LWPR: A Scalable Method for Incremental Online Learning in High Dimensions

Sethu Vijayakumar
sethu.vijayakumar@ed.ac.uk
School of Informatics, University of Edinburgh,
Edinburgh EH9 3JZ,United Kingdom
Aaron D'Souza, Stefan Schaal
{adsouza,sschaal}@usc.edu
Department of Computer Science, University of Southern California,
Los Angeles, CA 90089-2520,USA.

June 13, 2005

Abstract

Locally weighted projection regression (LWPR) is a new algorithm for incremental nonlinear function approximation in high dimensional spaces with redundant and irrelevant input dimensions. At its core, it employs nonparametric regression with locally linear models. In order to stay computationally efficient and numerically robust, each local model performs the regression analysis with a small number of univariate regressions in selected directions in input space in the spirit of partial least squares regression. We discuss when and how local learning techniques can successfully work in high dimensional spaces and compare various techniques for local dimensionality reduction before finally deriving the LWPR algorithm. The properties of LWPR are that it i) learns rapidly with second order learning methods based on incremental training, ii) uses statistically sound stochastic leave-one-out cross validation for learning without the need to memorize training data, iii) adjusts its weighting kernels based only on local information in order to minimize the danger of negative interference of incremental learning, iv) has a computational complexity that is linear in the number of inputs, and v) can deal with a large number of - possibly redundant - inputs, as shown in various empirical evaluations with up to 50 dimensional data sets. For a probabilistic interpretation, predictive variance and confidence intervals are derived. To our knowledge, LWPR is the first truly incremental spatially localized learning method that can successfully and efficiently operate in very high dimensional spaces.

1 Introduction

Despite the recent progress in statistical learning, nonlinear function approximation with high dimensional input data remains a nontrivial problem, especially in incremental and real-time formulations. There is, however, an increasing number of problem domains where both these properties are important. Examples include the on-line modeling of dynamic processes observed by visual surveillance, user modeling for advanced computer interfaces and game playing, and the learning of value functions, policies, and models for learning control, particularly in the context of high-dimensional movement systems like humans or humanoid robots. An ideal algorithm for such tasks needs to avoid potential numerical problems from redundancy in the input data, eliminate irrelevant input

dimensions, keep the computational complexity of learning updates low while remaining data efficient, allow for online incremental learning, and, of course, achieve accurate function approximation and adequate generalization.

When looking for a learning framework to address these goals, one can identify two broad classes of function approximation methods: (i) methods which fit non-linear functions *globally*, typically by input space expansions with predefined and/or parametrized basis functions and subsequent linear combinations of the expanded inputs, and (ii) methods which fit non-linear functions *locally*, usually by using spatially localized simple (e.g., low order polynomial) models in the original input space and automatically adjusting the complexity (e.g., number of local models and their locality) to accurately account for the nonlinearities and distributions of the target function. Interestingly, the current trends in statistical learning have concentrated on methods that fall primarily in the first class of global nonlinear function approximators, for example, Gaussian Process Regression(GPR)(Williams & Rasmussen, 1996), Support Vector Machine Regression(SVMR)(Smola & Scholkopf, 1998) and Variational Bayes for mixture models[1](VBM) (Ghahramani & Beal, 2000). In spite of the solid theoretical foundations that these approaches possess in terms of generalization and convergence, they are not necessarily the most suitable for on-line learning in high-dimensional spaces. First, they require an a priori determination of the right modeling biases. For instance, in the case of GPR and SVMR, these biases involve selecting the right function space in terms of the choice of basis or kernel functions(Vijayakumar & Ogawa, 1999), and in VBM the biases are concerned with the the right number of latent variables and proper initialization [2]. Second, all these recent function approximator methods were developed primarily for batch data analysis and are not easily and/or efficiently adjusted for incrementally arriving data. For instance, in SVMR, adding a new data point can drastically change the outcome of the global optimization problem in terms of which data points actually become support vectors, such that all (or a carefully selected subset) of data has to be kept in memory for re-evaluation. Thus, adding a new data point in SVMR is computationally rather expensive, a property that is also shared by GPR. VBM suffers from similar problems due to the need of storing and re-evaluating data when adding new mixture components (Ueda & Hinton, 2000). In general, it seems that most suggested Bayesian learning algorithms are computationally too expensive for real-time learning because they tend to represent the complete joint distribution of the data, albeit as a conditionally independent factored representation. As a last point, incremental approximation of functions with global methods is prone to lead to negative interference when input distributions change (Schaal & Atkeson, 1998) – such changes are, however, a typical scenario in many on-line learning tasks.

In contrast to the global learning methods described above, function approximation with spatially localized models are rather well suited for incremental and real-time learning, particularly in the framework of nonparametric regression (Atkeson, Moore, & Schaal, 1997). Nonparametric [3] methods are very useful when there is limited knowledge about the model complexity such that the model resources need to be increased in a purely incremental and data driven fashion, as demonstrated in previous work (Schaal & Atkeson, 1998). However, since these techniques allocate resources to cover the input space in a localized fashion, in general, with an increasing number of input dimensions, they encounter an exponential explosion in the number of local models required for accurate approximation, ofter referred to as the 'curse of dimensionality' (Scott, 1992). Hence, at the outset, high-dimensional function approximation seems to be computationally infeasible for

[1]Mixture models are actually somehow in between global and local function approximators since they use local model fitting but employ a global optimization criterion.

[2]It must be noted that there has been some recent work(Scholkopf & Smola, 1999) that have started to look at model selection for SVMs and GPRs and automatic determination of number of latent models for VBM (Ghahramani & Beal, 2000)

[3]It should be noted that "nonparametric regression" does not mean that there are no parameters in the algorithm, but rather that the underlying distributions of a learning problem cannot be indexed with a finite number of parameters. Thus, the number of parameters in nonparametric regression usually grows with the number of training data. GPR and SVMR are also nonparametric algorithms, just that they fall in the class of "global" methods as described above.

local nonparametric learning.

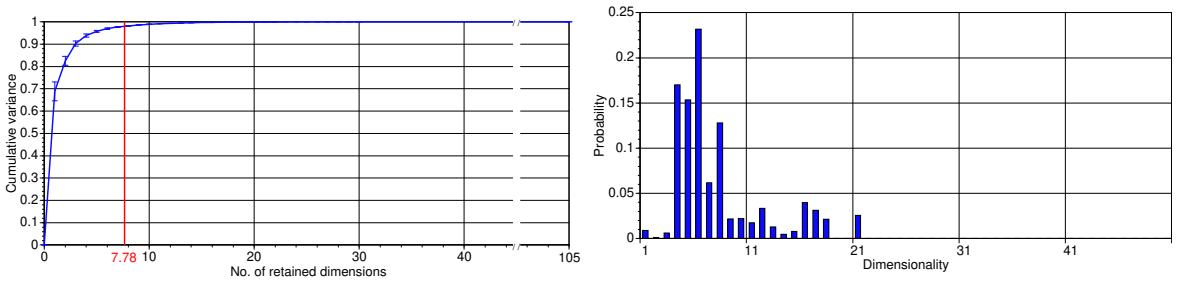
Nonparametric learning in high dimensional spaces with *global* methods, however, has been employed successfully by using techniques of projection regression (PR). PR copes with high dimensional inputs by decomposing multivariate regressions into a superposition of single variate regressions along a few selected projections in input space. The major difficulty of PR lies in the selection of efficient projections, i.e., how to achieve the best fitting result with as few univariate regressions as possible. Among the best known PR algorithms are projection pursuit regression (Friedman & Stutzle, 1981), and its generalization in form of Generalized Additive Models (Hastie & Tibshirani, 1990). Sigmoidal neural networks can equally be conceived of as a method of projection regression, in particular when new projections are added sequentially, e.g., as in Cascade Correlation (Fahlman & Lebiere, 1990).

In this paper we will suggest a method of extending the beneficial properties of *local nonparametric learning* to high dimensional function approximation problems. The prerequisite of our approach is that the high dimensional learning problems we address have locally low dimensional distributions, an assumption that holds for a large class of real world data (see below). If distributions are locally low dimensional, the allocation of local models can be restricted to these thin distributions, and only a tiny part of the entire high dimensional space needs to be filled with local models. Thus, the curse of dimensionality of spatially localized model fitting can be avoided. Under these circumstances, an alternative method of projection regression can be derived, focusing on finding efficient *local* projections. Local projections can be used to accomplish local function approximation in the neighborhood of a given query point with traditional local nonparametric approaches, thus inheriting most of the statistical properties from the well established methods of locally weighted learning and nonparametric regression (Hastie & Loader, 1993; Atkeson et al., 1997). As this paper will demonstrate, the resulting learning algorithm combines the fast, efficient and incremental capabilities of the non-parametric techniques while alleviating the problems faced due to high dimensional input domains through local projections.

In the following sections, we will first motivate why many high dimensional learning problems have locally low dimensional data distributions such that the prerequisites of our local learning system are justified. Second, we will address the question of how to find good local projections by looking into various schemes for performing dimensionality reduction for regression, including principal component regression, factor analysis, partial least squares regression, and covariance projection regression – a novel method that generalizes principal component regression towards partial least squares regression. Empirical evaluations will highlight the pros and cons of the different methods. Finally, we will embed the most efficient and robust of these projection algorithms in an incremental nonlinear function approximator (Vijayakumar & Schaal, 1998) capable of automatically adjusting the model complexity in a purely data driven fashion. In several evaluations - both on synthetic and real world data - the resulting incremental learning system demonstrates high accuracy for function fitting in very high dimensional spaces, robustness towards irrelevant and redundant inputs, as well as low computational complexity. Comparisons will prove the competitiveness with other state-of-the-art learning systems.

2 Evidence for low dimensional distributions

The development of our learning system in the next sections relies on the assumption that high dimensional data sets have locally low dimensional distributions, an assumption that requires some clarification. Across domains like vision, speech, motor control, climate patterns, human gene distributions, and a range of other physical and biological sciences, various researchers have reported evidence that corroborate the fact that the true intrinsic dimensionality of high dimensional data is often very low (Tenenbaum & Langford, 2000; Roweis & Saul, 2000; Vlassis, 2002). We interpret these findings as evidence that the physical world has a significant amount of coherent structure that expresses itself in terms of strong correlations between different variables that describe the state



(a) (b)

Figure 1: Dimensionality Analysis - (a) The cumulative variance accounted versus the local dimensionality (avearged across all mixture models) (b) The distribution of the effective dimensionality across all mixture models

of the world at a particular moment in time. For instance, in computer vision it is quite obvious that neighboring pixels of an image of a natural scene have redundant information. Moreover, the probability distribution of natural scenes in general has been found to be highly structured such that it lends itself to a sparse encoding in terms of set of basis functions (Olshausen & Field, 1996; Bell & Sejnowski, 1997). Another example comes from our own research on human motor control. Despite that humans can accomplish movement tasks in almost arbitrary ways – thus possibly generating arbitrary distributions of the variables that describe their movements – behavioral research has discovered a tremendous amount of regularity within and across individuals (Kawato, 1999; Schaal & Sternad, 2001). These regularities lead to locally low dimensional data distribution, as illustrated in the example in Fig. 1. In this analysis(D'Souza & Schaal, 2001), we assessed the intrinsic dimensionality of data collected from full body movement of several human subjects, collected with a special full-body exoskeleton that recorded simultaneously 35 degrees-of-freedom of the joint angular movement of the subjects at 100hz sampling frequency. Subjects performed a variety of daily-life tasks (e.g., walking, object manipulation, reaching, etc.) until about a gigabyte of data was accumulated. Our analysis examined the local dimensionality of the joint distribution of positions, velocities, and accelerations of the collected data, i.e., a 105 dimensional data set, as would be needed as inputs to learn an inverse dynamics model for motor control (Kawato, 1999). As an analysis tool, we employed a variational Bayesian mixture of factor analyzers that automatically estimated the required number of mixture components (Ghahramani & Beal, 2000). As demonstrated in Fig. 1(a), the local dimensionality was around 5 to 8 dimensions, computed based on the average number of significant latent variables per mixture component. Fig. 1(b) shows the distribution of the effective dimensionality across all mixture models.

In summary, the results from our analysis and other sources in the literature show that there is a large class of high dimensional problems that can be treated locally in much lower dimensions if one can determine appropriate regions of locality and the local projections that model the corresponding low dimensional distributions. As a caveat, however, it may happen that such low dimensional distributions are embedded in additional dimensions that are irrelevant for the problem at hand but have considerable variance. In the context of regression, it will thus be important to only model those local dimensions that carry information that is important for the regression and eliminate all other dimensions, i.e., to perform local dimensionality reduction with regression in mind and not just based on input or joint input-output distributions, as discussed in the next section.

3 Local dimensionality reduction

Assuming that data is characterized by locally low dimensional distributions, efficient algorithms are needed to exploit this property. For this purpose, we will focus on locally weighted learning (LWL) methods (Atkeson et al., 1997) because they allow us to adapt a variety of linear dimensionality reduction techniques for the purpose of nonlinear function approximation (see Section 4) and because they are easily modified for incremental learning. LWL-related methods have also found widespread application in mixture models (Jordan & Jacobs, 1994; Xu, Jordan, & Hinton, 1995; Ghahramani & Beal, 2000) such that the results of this section can contribute to this field, too.

In pursuit of the question of what is the "right" method to perform local dimensionality reduction for regression, the following sections will compare several candidate techniques theoretically and empirically. In particular, we will derive and investigate locally weighted versions of principal component regression (LWPCR), joint data principal component analysis (LWPCA), factor analysis (LWFA) and partial least squares (LWPLS). In addition, we will develop a novel family of projection regression technique, named covariant projection regression (LWCPR). In the next sections, we will first outline the algorithms of each of these methods before conducting empirical evaluations to test their robustness in situations where their data generating model is violated.

3.1 The locally weighted regression model

The learning problems considered here assume the standard regression model:

$$y = f(\mathbf{x}) + \epsilon.$$

where $\mathbf{x}$ denotes the d-dimensional input vector, y the (for simplicity) scalar output, and ϵ a mean-zero random noise term. When only a local subset of data in the vicinity of a point $\mathbf{x}_c$ is considered and the locality is chosen appropriately, a low order polynomial can be employed to model this local subset. Due to a favorable compromise between computational complexity and quality of function approximation (Hastie & Loader, 1993), we choose linear models

$$y = \boldsymbol{\beta}^T \mathbf{x} + \epsilon.$$

A measure of locality for each data point, the weight w_i, is computed from a Gaussian kernel:

$$w_i = exp(-0.5(\mathbf{x}_i - \mathbf{x}_c)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_c)), \quad \text{and} \quad \mathbf{W} \equiv diag\{w_1, \ldots, w_M\} \tag{1}$$

where $\mathbf{D}$ is a positive semi-definite distance metric which determines the size and shape of the neighborhood contributing to the local model (Atkeson et al., 1997). The weights w_i will enter all following algorithms to assure spatial localization in input space. In particular, zero mean prerequisites of several algorithms is ensured by subtracting the weighted mean $\overline{\mathbf{x}}$ or $\overline{y}$ from the data, where

$$\overline{\mathbf{x}} = \sum_{i=1}^{M} w_i \mathbf{x}_i / \sum_{i=1}^{M} w_i, \text{ and } \overline{y} = \sum_{i=1}^{M} w_i y_i / \sum_{i=1}^{M} w_i, \tag{2}$$

and M denotes the number of data points. For simplicity, and without loss of generality, we will thus assume in the derivations in the next sections that all input and output data have a weighted mean of zero with respect to a particular weighting kernel. The input data is summarized in the rows of the matrix $\mathbf{X}=[\mathbf{x}_1\ \mathbf{x}_2\ ...\mathbf{x}_M]^T$, the corresponding outputs are the coefficients of the vector $\mathbf{y}$, and the corresponding weights, determined from eq.(1), are in the diagonal matrix $\mathbf{W}$. In some cases, we need the joint input and output data, denoted as $\mathbf{Z}=[\mathbf{z}_1\ \mathbf{z}_2...\mathbf{z}_M]^T= [\mathbf{X}\ \mathbf{y}]$.

3.2 Locally Weighted Factor Analysis (LWFA)

Factor analysis(Everitt, 1984) is a density estimation technique that assumes that the observed data $\mathbf{z}$ were actually generated from a lower dimensional process, characterized by k *latent* or *hidden*

variables $\mathbf{v}$ that are all independently distributed with mean zero and unit variance. The observed variables are generated form the latent variables through the transformation matrix $\mathbf{U}$ and additive mean zero independent noise $\boldsymbol{\epsilon}$ with diagonal covariance matrix $\boldsymbol{\Omega}$:

$$\mathbf{z} = \mathbf{U}\mathbf{v} + \boldsymbol{\epsilon}, \tag{3}$$

where

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \boldsymbol{\epsilon}_x \\ \epsilon_y \end{bmatrix}, E\{\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\} = \boldsymbol{\Omega}, \tag{4}$$

and $E\{.\}$ denotes the expectation operator. If both $\mathbf{v}$ and $\boldsymbol{\epsilon}$ are normally distributed, the parameters $\boldsymbol{\Omega}$ and $\mathbf{U}$ can be obtained iteratively by the Expectation-Maximization algorithm (EM) (Rubin & Thayer, 1982).

Factor analysis can be adapted for linear regression problems by assuming that $\mathbf{z}$ was generated with

$$\mathbf{U} = [\mathbf{I}^d \ \boldsymbol{\beta}]^T \tag{5}$$

$$\mathbf{v} = \mathbf{x} \tag{6}$$

where $\boldsymbol{\beta}$ denotes the vector of regression coefficients of the linear model $y = \boldsymbol{\beta}^T\mathbf{x}$ and $\mathbf{I}^d$ the d-dimensional identity matrix. For the standard regression model, $\boldsymbol{\epsilon}_x$ would be zero, i.e. we consider noise contamination in the output only – for numerical reasons, however, some remaining variance needs to be permitted and we prefer to leave it to the EM algorithm to find the optimal values of the covariance $\boldsymbol{\Omega}$. After calculating $\boldsymbol{\Omega}$ and $\mathbf{U}$ with EM in joint data space as formulated in eq.(3), an estimate of $\boldsymbol{\beta}$ can be derived from the conditional probability $p(y|\mathbf{x})$. Let us denote $\mathbf{Z}=[\mathbf{z}_1 \ \mathbf{z}_2 ... \mathbf{z}_M]^T$ and $\mathbf{V}=[\mathbf{v}_1 \ \mathbf{v}_2 ... \mathbf{v}_M]^T$. The locally weighted version (LWFA) of $\boldsymbol{\beta}$ can be obtained together with an estimate of the factors $\mathbf{v}$ from the joint weighted covariance matrix $\boldsymbol{\Psi}$ of $\mathbf{z}$ and $\mathbf{v}$ as

$$E\{\begin{bmatrix} y \\ \mathbf{v} \end{bmatrix} |\mathbf{x}\} = \begin{bmatrix} \boldsymbol{\beta}^T \\ \mathbf{B} \end{bmatrix} \mathbf{x} = \Psi_{21}\Psi_{11}^{-1}\mathbf{x}, \tag{7}$$

where

$$\boldsymbol{\Psi} = [\mathbf{Z}^T \ \mathbf{V}^T]\mathbf{W} \begin{bmatrix} \mathbf{Z} \\ \mathbf{V} \end{bmatrix} / \sum_{i=1}^{M} w_i = \begin{bmatrix} \Omega + \mathbf{U}\mathbf{U}^T & \mathbf{U} \\ \mathbf{U}^T & \mathbf{I}^d \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix}, \tag{8}$$

and $\mathbf{B}$ is a matrix of coefficients involved in estimating the factors $\mathbf{v}$. Note that unless the noise $\boldsymbol{\epsilon}_x$ is zero, the estimated $\boldsymbol{\beta}$ is different from the true $\boldsymbol{\beta}$ as it tries to optimally average out the noise in the data. Thus, factor analysis can also be conceived of as a tool for linear regression with noise contaminated inputs.

3.3 Principal Component Regression(LWPCR)

Although not optimal, a computationally efficient technique of dimensionality reduction for linear regression is the principal component regression (Massy, 1965; Vijayakumar & Schaal, 1998). For locally weighted principal component regression (LWPCR), let us represent the covariance matrix of the weighted input data, $\mathbf{C}^{lwpcr}$, as

$$\mathbf{C}^{lwpcr} = \sum_{i=1}^{M} w_i \mathbf{x}_i \mathbf{x}_i^T / \sum_{i=1}^{M} w_i \tag{9}$$

and its eigenvalue decomposition in form of

$$\mathbf{C}^{lwpcr}\mathbf{c}_i = \lambda_i \mathbf{c}_i \tag{10}$$

where, $\lambda_1 > \lambda_2 > ....$ The dimensionality reduction step in LWPCR projects the inputs onto the k largest principal components of $\mathbf{C}^{lwpcr}$ by the transformation matrix $\mathbf{U}$ whose columns are the unit eigenvectors corresponding to the k largest eigenvalues of $\mathbf{C}^{lwpcr}$ given by

$$\mathbf{U} \equiv [\mathbf{c}_1 \ \mathbf{c}_2 \ ... \ \mathbf{c}_k]. \tag{11}$$

The vector of regression coefficients can then be calculated as

$$\boldsymbol{\beta} = (\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{W}y. \tag{12}$$

Eq.(12) is inexpensive to evaluate since after projecting $\mathbf{X}$ with $\mathbf{U}$, $\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U}$ becomes a diagonal matrix that is can be inverted easily. Essentially, LWPCR performs univariate regressions along orthogonal projection directions of the inputs space. A pseudocode implementation of the LWPCR is given in the Appendix A.

LWPCR assumes that the inputs have additive spherical noise, which includes the zero noise case. Since, during dimensionality reduction, LWPCR does not take into account the output data, it is susceptible to ignoring input dimensions with low variance which nevertheless have important contribution to the regression output. Also, if the input data has non-spherical noise and irrelevant dimensions, LWPCR is prone to focus the regression on irrelevant projections.

3.4 Principal Component in Joint Space(LWPCA_J)

An alternative way of determining the parameters in a reduced space employs locally weighted principal component analysis in the joint data space (LWPCA_J). Let us denote the weighted covariance matrix of the joint data $\mathbf{Z} = [\mathbf{X}\mathbf{y}]$ as $\mathbf{C}^{lwpca}$, which is given by

$$\mathbf{C}^{lwpca} = \sum_{i=1}^{M} w_i(\mathbf{z}_i - \overline{\mathbf{z}})(\mathbf{z}_i - \overline{\mathbf{z}})^T / \sum_{i=1}^{M} w_i. \tag{13}$$

Let the eigenvalue decomposition of $\mathbf{C}^{lwpca}$ be given as

$$\mathbf{C}^{lwpca}\mathbf{c}_i = \lambda_i\mathbf{c}_i \tag{14}$$

where, $\lambda_1 > \lambda_2 > ....$ Let us define a dimensionality reduction transformation matrix $\mathbf{U}$ as

$$\mathbf{U} \equiv [\mathbf{c}_1 \ \mathbf{c}_2 \ ... \ \mathbf{c}_{k+1}], \tag{15}$$

which is the projection onto the space spanned by the $k+1$ largest principal components of the joint data. Noting that the eigenvectors in $\mathbf{U}$ are unit length, the matrix inversion theorem (Horn & Johnson, 1994) provides a means to derive an efficient estimate of $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta} = \mathbf{U}_x(\mathbf{U}_y^T - \mathbf{U}_y^T(\mathbf{U}_y\mathbf{U}_y^T - \mathbf{I}^p)^{-1}\mathbf{U}_y\mathbf{U}_y^T), \tag{16}$$

where $\mathbf{U}$ is decomposed into $\mathbf{U}_x$ and $\mathbf{U}_y$ as

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix}, \tag{17}$$

and p is the dimensionality of the outputs. In our one dimensional output case, $p = 1$ and $\mathbf{U}_y$ is just a $(1 \times k)$-dimensional row vector, and the evaluation of eq.(16) does not require a matrix inversion anymore but rather only a division. If we assume normal distributions in all variables as in LWFA, LWPCA_J is the special case of LWFA where the noise covariance is spherical, i.e., the same magnitude of noise in all observables. Under these circumstances, the subspaces spanned by $\mathbf{U}$ in both methods will be the same (Tipping & Bishop, 1999). However, the regression coefficients of LWPCA_J will be different from those of LWFA unless the noise level is zero, as LWFA optimizes the coefficients by taking into account the noise in the data (refer eq.(7)). Thus, for normal distributions and a correct guess of k, LWPCA_J is always expected to perform worse than LWFA.

Table 1: Outline of Partial Least Squares (PLS) Regression Algorithm

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$
2. **for** $i = 1$ **to** k **do**
   (a) $\mathbf{u}_i = \mathbf{X}_{res}^T \mathbf{y}_{res}$.
   (b) $\beta_i = \mathbf{s}_i^T \mathbf{y}_{res} / (\mathbf{s}_i^T \mathbf{s}_i)$ where $\mathbf{s}_i = \mathbf{X}_{res} \mathbf{u}_i$.
   (c) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{s}_i \beta_i$, $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{s}_i {\mathbf{p}_i}^T$ where $\mathbf{p}_i = \mathbf{X}_{res}^T \mathbf{s}_i / (\mathbf{s}_i^T \mathbf{s}_i)$.

3.5 Partial Least Squares(LWPLS)

Partial least squares (PLS) (Wold, 1975; Frank, 1993), a technique extensively used in chemometrics, recursively computes orthogonal projections of the input data and performs single variable regressions along these projections on the residuals of the previous iteration step. Table 1 provides an outline of the PLS algorithm. The key ingredient in PLS is to use the direction of maximal correlation between the residual error and the input data as the projection direction at every regression step. Additionally, PLS regresses the inputs of the previous step against the projected inputs $\mathbf{s}$ in order to ensure the orthogonality of all the projections $\mathbf{u}$ (Step 2c). Actually, this additional regression could be avoided by replacing $\mathbf{p}$ with $\mathbf{u}$ in Step 2c, similar to techniques used in principal component analysis (Sanger, 1989). However, using this regression step leads to better performance of the algorithm as PLS chooses the most effective projections if the input data has a spherical distribution: in the spherical case, with only one projection, PLS will find the direction of the gradient and achieve optimal regression results. The regression step in 2c chooses the reduced input data $\mathbf{X}_{res}$ such that the resulting data vectors have minimal norms and, hence, push the distribution of $\mathbf{X}_{res}$ to become more spherical. An additional consequence of 2c is that all the projections $\mathbf{s}_i$ become uncorrelated, i.e., $\mathbf{s}_j^T \mathbf{s}_i = 0 \; \forall i \neq j$; a property which will be important in the derivations below. Appendix A provides the modifications necessary for implementing a locally weighted version (LWPLS) of the algorithm.

3.6 Covariant Projection Regression (LWCPR)

In this section, we introduce a new algorithm which has the flavor of both PCR and PLS. Covariant Projection Regression(CPR) transforms the input data by a (diagonal) weight matrix $\mathbf{T}$

$$\mathbf{X}_t = \mathbf{T}\mathbf{X}$$

with the goal to elongate the distribution in the direction of the gradient of the input/output relation of the data. Subsequently, the major principal component of this deformed distribution is chosen as the direction of a univariate regression. In contrast to PCR, this projection now reflects not only the influence of the input distribution but also that of the regression outputs. As in PLS, if the input distribution is spherical, CPR will obtain an optimal regression result with a *single* univariate fit, irrespective of the actual input dimensionality. If the input distribution is not spherical, several CPR projection need to be cascaded, similar to PLS. Refer to Appendix A for the implementation pseudocode.

CPR is actually a family of algorithms, depending on how the weights in $\mathbf{T}$ are chosen. We consider two such options:

Weighting scheme CPR1: $T_{ii}^{cpr1} = \frac{1}{\sigma_{\mathbf{x}_{res}}} \left(\frac{|y_{res,i}|}{\|\mathbf{x}_{res,i}\|} \right)^p$.

Weighting scheme CPR2: $T_{ii}^{cpr2} = \frac{1}{\|\mathbf{x}_{res,i}\|} \left(\frac{|y_{res,i}|}{\|\mathbf{x}_{res,i}\|} \right)^p$.

Note that the subscript *res* above denotes that $\mathbf{x}$ and y may not be the original input and output data anymore but rather residuals from the previous projection.

CPR1 spheres the input data and then weights it by the "slope" of each data point, taken to the power p (where p is an even number) for increasing the impact of the input/output relationships. CPR2 is a variant that, although a bit idiosyncratic, had the best average performance in practice: CPR2 first maps the input data onto a unit-(hyper)sphere, and then stretches the distribution along the direction of maximal slope, i.e., the regression direction – this method is fairly insensitive to noise in the input data.

3.7 Monte Carlo evaluations for performance comparison

3.7.1 Methods

In order to evaluate the candidate algorithms empirically, mean zero linear data sets, consisting of 1000 training points and 2000 test points, with 5 input dimension and 1 output were generated at random. The outputs were calculated from random linear coefficients, and Gaussian noise was added. Then, the input data were augmented with 5 additional dimensions containing only zero values and rotated and stretched to have random variances in all dimensions. For some test cases, 5 more input dimensions with random (non-constant) values were added afterwards to explore the effect of irrelevant inputs on the regression performance. Empirically, we determined $p = 4$ as a good choice for the CPR methods. Given that the datasets were perfectly linear, we omitted any local weighting of the data, i.e., $w_i = 1$ for all data points.

The simulations considered the following permutations:

- Low noise[4] (r^2=0.99) and High noise ($r^2 = 0.9$) in output data.
- With and without irrelevant (non-constant) input dimensions.
- With four different guesses for the dimensionality of the latent variable space, i.e., $k = 1, 4, 5,$ or 6.

Each algorithm was run[5] 100 times on random data sets of each of the 16 combinations of test conditions. The nMSE results of all trials were averaged and are reported as "averaged nMSE". Fig. 2 shows the summary results. The best nMSE values possible under the chosen noise conditions are nMSE=0.01 for the low noise cases and nMSE=0.1 for the high noise case.

3.7.2 Results

Using only one projection, i.e., $k = 1$, was intended to investigate how well each of the algorithms would be suitable for insertion into a learning scheme that increases the number of projections incrementally. For such constructive learning, it is desirable that the first projection itself reduces the mean squared error of the regression as much as possible. With respect to this criterion, Fig.2(a) demonstrates that LWFA and LWPCR perform rather bad, while LWPLS, LWCPR and LWPCA_J achieve significantly better results. For LWFA, the reduced performance is most likely to be attributed to overestimating the noise in the data, which subsequently adversely affects the determination of the regression parameters. For LWPCR, the performance disadvantage can be blamed on selecting the projection direction without taking the outputs into account. In contrast,

[4]Noise is parametrized by the coefficient of determination r^2 of standard linear regression, defined as $r^2 = ({\sigma^2}_y - {\sigma^2}_{res})/{\sigma^2}_y$, where ${\sigma^2}_{res}$ is the variance of the residual error and ${\sigma^2}_y$ is the variance of the outputs. We add noise scaled to the variance of the non-noise contaminated outputs $\check{y}$ such that ${\sigma^2}_{noise} = c{\sigma^2}_{\check{y}}$, where $c = \frac{1}{r2} - 1$. The best normalized mean squared error (nMSE) achievable by a learning system under this noise level is $1 - r^2$, unless the system overfits the data.

[5]Except for LWFA, all methods can evaluate a data set in non-iterative calculations. LWFA was trained with EM for maximally 1000 iterations or until the log-likelihood increased less than 1.0e-10 in one iteration.

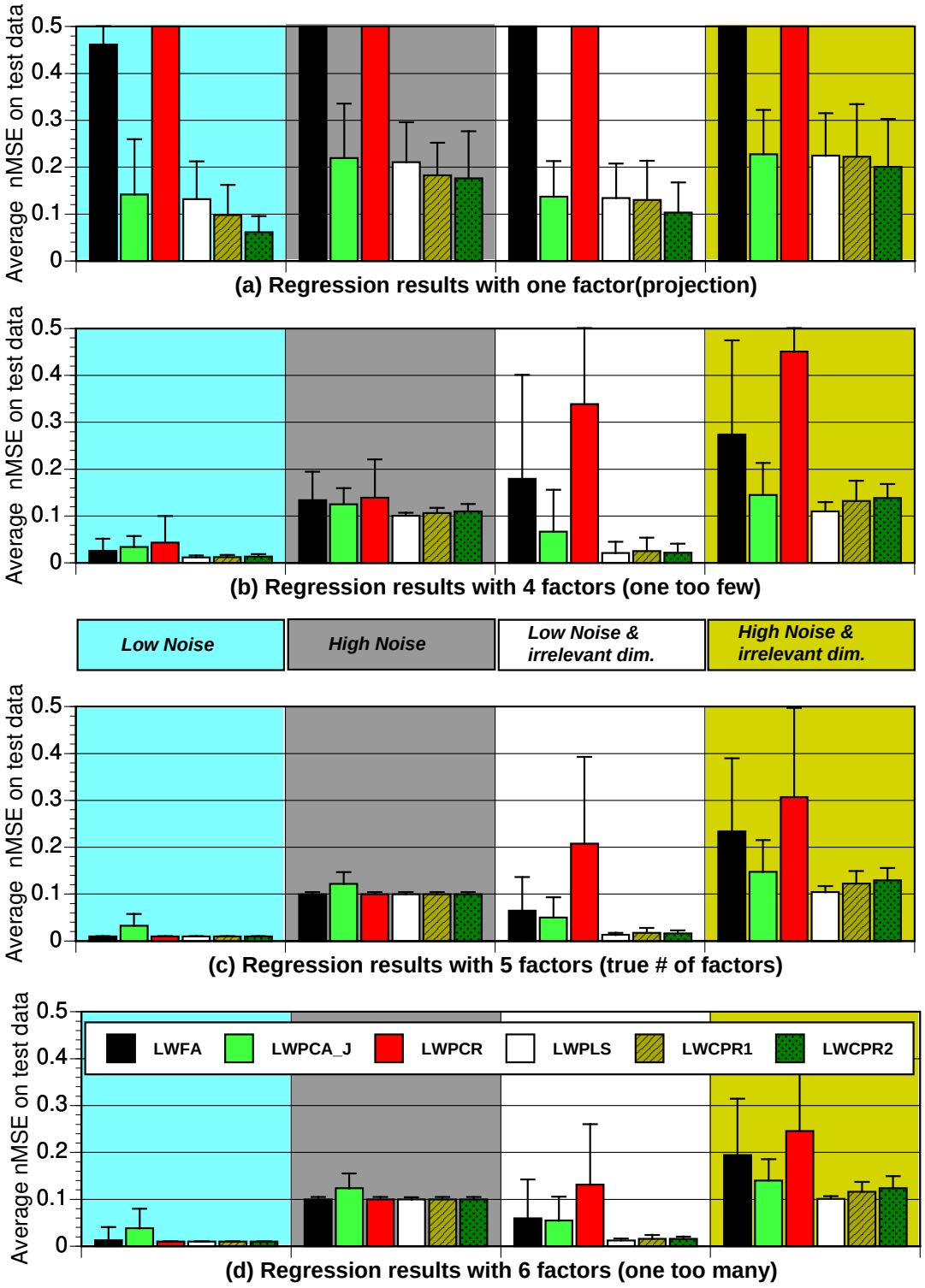


Figure 2: Summary results of Monte Carlo experiments. The subplots show results for projection dimensions 1, 4, 5 and 6. Each of the subplots has four additional conditions made of permutations of: (i) low and high noise (ii) with and without irrelevant (non constant) inputs. Error bars are one standard deviation computed from 100 repetitions in each test condition. The best nMSE values achievable are 0.01 (for the low noise datasets) and 0.1 (for the high noise datasets).

the slightly heuristic determination of projection directions in LWPLS and LWCPR proves to be rather powerful, as previously observed in numerous PLS papers (Frank, 1993).

In Fig.2(c), the number of factors equals the underlying dimensionality of the problem. All algorithms are essentially performing equally well in the case of no irrelevant dimensions, although LWPCA_J shows some disadvantage due to its assumption of equal amounts of noise in *both* inputs and outputs. However, with five extra irrelevant inputs, i.e., a total number of 15 input dimensions with underlying latent space of 10 dimensions, LWFA and LWPCR also show significant degradation in performance. For LWPCR, this effect can again be attributed to its dimensionality reduction based on only the inputs. For LWFA, it is important that the entire latent space is represented by the factors, even if some of these variables are irrelevant for regression. Thus, the same effect as explained for $k = 1$ accounts for LWFA's reduced performance.

For the case of underestimating the number of necessary projections by one (Fig.2(b)), again LWPCR and LWFA show reduced performance for the same reasons as for $k = 5$ with irrelevant inputs. LWPLS and LWCPR do rather well with LWPLS having a slight lead in noisier situations. The advantage of LWPLS for situations with more noise is due to its method of projection selection: noise is averaged out in the correlation of inputs and outputs, while in LWCPR, noise can affect the selection of the projection direction as it is nonlinearly transformed by the exponent p. LWPCA_J ranks third in this comparison.

Finally, for the case with one factor too many (Fig.2(d)), the pattern of $k = 5$ repeats itself. Thus, all methods show no significant degradation due to using too many projections. However, in practice, we will be more interested in approximations with a smaller number of projections than actually needed in order to save resources and reduce computational complexity.

We conclude that LWPLS and LWCPR outperform LWPCR, LWFA and LWPCA_J by a significant margin when the learning data does not match the assumptions of the chosen algorithm. LWPCR tends to deteriorate due to performing dimensionality reduction without taking the regression outputs into account, thus rejecting dimensions that have low variance but are nevertheless important for reducing the mean squared error. Although being theoretically the most appealing, LWFA turned out to be very sensitive to the selection of the correct number of factors, essentially requiring that the entire latent space of the inputs is represented, no matter whether some of these dimensions contribute to the regression or not. This property makes LWFA rather expensive for high dimensional problems and also less suitable for growing the number of projections incrementally. LWPCA_J turned out to be a mediocre but on average, a robust method of dimensionality reduction. In all situations, LWCPR and LWPLS performed equally good or better than any of the remaining algorithms. For low noise ($r^2 = 0.99$), LWCPR has a marginal lead over LWPLS, especially during the first projections. For high noise cases ($r^2 = 0.9$), LWPLS has a slight advantage, due to its noise insensitive way of selecting projections. Among the LWCPR candidates, LWCPR2 seems to outperform LWCPR1 in low noise cases, while this trend is reversed for high noise cases – the difference between the two algorithms is statistically insignificant.

In summary, both LWCPR and LWPLS perform very well and accomplish excellent regression results with relatively few projections since their choice of projection directions does not just try to take into account the input distribution but also the gradient of the data. In the next section, we will embed one of these algorithms in an incremental nonlinear function approximator to demonstrate its abilities for non-trivial function fitting problems.

4 Locally Weighted Projection Regression

For nonlinear function approximation, the core concept of our learning system – Locally Weighted Projection Regressionn (LWPR), is to find approximations by means of piecewise linear models (Atkeson et al., 1997). Learning involves automatically determining the appropriate *number* of local models K, the parameters $\boldsymbol{\beta}_k$ of the *hyperplane* in each model, and also the *region of validity*,

Table 2: Legend of indexes and symbols used for LWPR

Notation	Affectation
M	No. of training data points
N	Input dimensionality (i.e. dim. of $\mathbf{x}$)
$k = (1 : K)$	Number of local models
$r = (1 : R)$	Number of local projections used by PLS
$\{\mathbf{x}_i, y_i\}_{i=1}^M$	Training Data
$\{\mathbf{z}_i\}_{i=1}^M$	Lower dimensional projection of input data $\mathbf{x}_i$(by PLS)
$\{z_{i,r}\}_{r=1}^R$	Elements of projected input
$\mathbf{X}$, $\mathbf{Z}$	Batch representations of input and projected data
w	Weight or activation of data $(\mathbf{x}, y)$ with respect to local model or receptive field(RF) center $\mathbf{c}$
$\mathbf{W}$	Weight matrix: $\mathbf{W} \equiv diag\{w_1, \ldots, w_N\}$
W^n	Cumulative weights (after n data points) seen be local model
$a_{var,r}^n$	Trace variable for incremental computation of r-th dimension of variable $'var'$ after seeing n data points.

called receptive field (RF), parameterized as a distance metric $\mathbf{D}_k$ in a Gaussian kernel:

$$w_k = exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x} - \mathbf{c}_k)). \tag{18}$$

Given a query point $\mathbf{x}$, every linear model calculates a prediction $\hat{y}_k(\mathbf{x})$. The total output of the learning system is the normalized weighted mean of all K linear models:

$$\hat{y} = \sum_{k=1}^{K} w_k \hat{y}_k / \sum_{k=1}^{K} w_k, \tag{19}$$

also illustrated in Fig. 3. The centers $\mathbf{c}_k$ of the RFs remain fixed in order to minimize negative interference during incremental learning that could occur due to changing input distributions (Schaal & Atkeson, 1998). Local models are createdon an 'as-needed' basis as described in Section 4.2. Table 2 provides a reference list of indices and symbols that are used consistently across the description of the LWPR algorithm.

4.1 Learning with LWPR

Despite its appealing simplicity, the "piecewise linear modeling" approach becomes numerically brittle and computationally too expensive in high dimensional input spaces when using ordinary linear regression to determine the local model parameters (Schaal & Atkeson, 1998). Given the empirical observation (cf. Section 2) that high dimensional data often lie on locally low dimensional distributions, and given the algorithmic results in Section 3, we will thus use local projection regression, i.e., LWPLS or LWCPR, within each local model to fit the hyperplane. As a significant computational advantage, we expect that far fewer projections than the actual number of input dimensions are needed for accurate learning. For an incremental learning algorithm, we chose LWPLS for our nonlinear implementation as PLS is already well established in the literature and is more straightforward to convert to incremental updates (cf. (Vijayakumar & Schaal, 1998) for

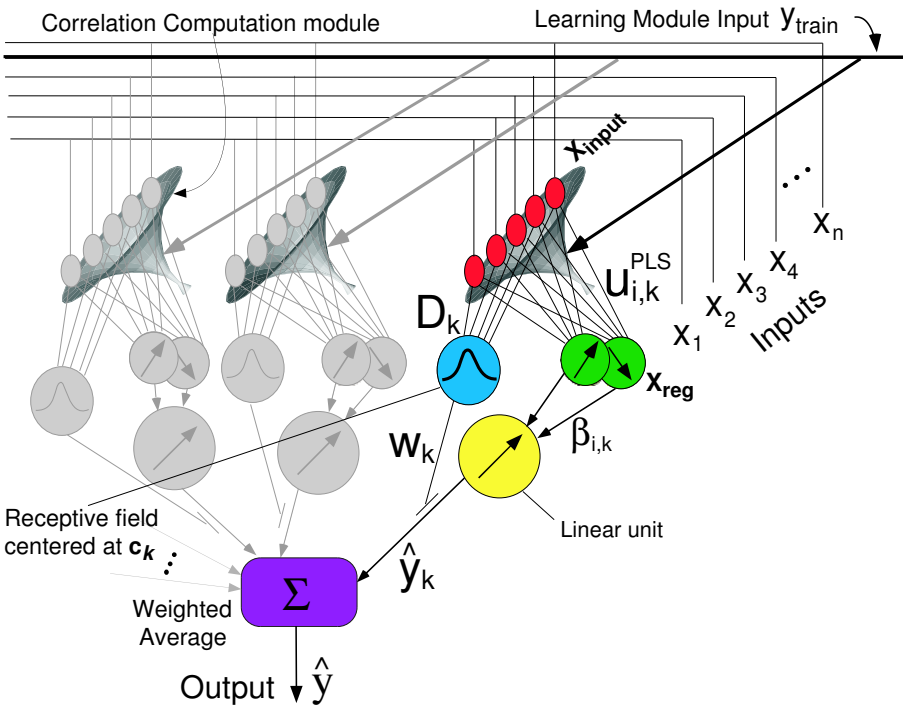


Figure 3: Information processing unit of LWPR

an incremental strategy for LWCPR). The next sections will describe the necessary modifications of LWPLS for this implementation, embed the local regression into the LWL framework, explain a method of automatic distance metric adaptation, and finish with a complete nonlinear learning scheme, called locally weighted projection regression (LWPR).

4.1.1 Incremental computation of projections and local regression

For incremental learning, i.e., a scheme that does not explicitly store any training data, the sufficient statistics of the learning algorithm need to be accumulated in appropriate variables. Table 3 (Vijayakumar & Schaal, 2000) provides suitable incremental update rules. The variables $a_{zz,r}$, $a_{zres,r}$ and $\mathbf{a}_{xz,r}$ are sufficient statistics that enable us to perform the univariate regressions in step 2c.1.2 and step 2c.2.2, similar to recursive least squares, i.e., a fast Newton-like incremental learning technique. $\lambda \in [0, 1]$ denotes a forgetting factor that allows exponential forgetting of older data in the sufficient statistics. Forgetting is necessary in incremental learning since a change of some learning parameters will affect a change in the sufficient statistics – such forgetting factors are a standard technique in recursive system identification (Ljung & Soderstrom, 1986). It can be shown that the prediction error of step 2b corresponds to the leave-one-out cross validation error of the current point *after* the regression parameters were updated with the data point - hence, it is denoted by e_{cv}.

In Table 3, for $R = N$, i.e. the same number of projections as the input dimensionality, the entire input space would be spanned by the projections $\mathbf{u}_r$ and the regression results would be identical to that of ordinary linear regression(Wold, 1975). However, once again, we would like to emphasize the important properties of the local projection scheme. First, if all the input variables are statistically independent and have equal variance[6], PLS will find the optimal projection direction $\mathbf{u}_r$ in roughly

[6]It should be noted that we could insert one more preprocessing step in Table 3 that independently scales all inputs

Table 3: Incremental locally weighted PLS for one RF centered at $\mathbf{c}$

---

1. Initialization: (# data points seen $n = 0$)
$\mathbf{x}_0^0 = \mathbf{0}$, $\beta_0^0 = 0$, $W^0 = 0$, $\mathbf{u}_r^0 = \mathbf{0}$; $\quad r = 1 : R$

2. Incorporating new data: Given training point $(\mathbf{x}, y)$

2a. Compute activation and update the means

1. $w = exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \mathbf{D}(\mathbf{x} - \mathbf{c}))$; $\quad W^{n+1} = \lambda W^n + w$
2. $\mathbf{x}_0^{n+1} = (\lambda W^n \mathbf{x}_0^n + w\mathbf{x})/W^{n+1}$; $\quad \beta_0^{n+1} = (\lambda W^n \beta_0^n + wy)/W^{n+1}$

2b. Compute the current prediction error

$\mathbf{x}_{res,1} = \mathbf{x} - \mathbf{x}_0^{n+1}$, $\hat{y} = \beta_0^{n+1}$

Repeat for $r = 1 : R$ (# projections)

1. $z_r = \mathbf{x}_{res,r}^T \mathbf{u}_r^n / \sqrt{\mathbf{u}_r^{nT} \mathbf{u}_r^n}$
2. $\hat{y} \leftarrow \hat{y} + \beta_r^n z_r$
3. $\mathbf{x}_{res,r+1} = \mathbf{x}_{res,r} - z_r \mathbf{p}_r^n$
4. $MSE_r^{n+1} = \lambda MSE_r^n + w\ (y - \hat{y})^2$

$e_{cv} = y - \hat{y}$

2c. Update the local model

$res_1 = y - \beta_0^{n+1}$

For $r = 1 : R$ (# projections)

2c.1 *Update the local regression and compute residuals*

1. $a_{zz,r}^{n+1} = \lambda\ a_{zz,r}^n + w\ z_r^2$; $\quad a_{zres,r}^{n+1} = \lambda\ a_{zres,r}^n + w\ z_r\ res_r$
2. $\beta_r^{n+1} = a_{zres,r}^{n+1} / a_{zz,r}^{n+1}$
3. $res_{r+1} = res_r - z_r \beta_r^{n+1}$
4. $\mathbf{a}_{xz,r}^{n+1} = \lambda\ \mathbf{a}_{xz,r}^n + w\mathbf{x}_{res,r} z_r$

2c.2 *Update the projection directions*

1. $\mathbf{u}_r^{n+1} = \lambda\ \mathbf{u}_r^n + w\mathbf{x}_{res,r}\ res_r$
2. $\mathbf{p}_r^{n+1} = \mathbf{a}_{xz,r}^{n+1} / a_{zz,r}^{n+1}$

$e = res_{r+1}$

3. Predicting with novel data ($\mathbf{x}_q$): Initialize: $y_q = \beta_0, \mathbf{x}_q = \mathbf{x}_q - \mathbf{x}_0$

Repeat for $r = 1 : R$

- $y_q \leftarrow y_q + \beta_r s_r$ where $s_r = \mathbf{u}_r^T \mathbf{x}_q$
- $\mathbf{x}_q \leftarrow \mathbf{x}_q - s_r \mathbf{p}_r^n$

Note: The subscript k referring to the k-th local model is omitted throughout this table since we are referring to updates in one local model or RF

---

a single sweep through the training data – the optimal projection direction corresponds to the gradient of the local linearization parameters of the function to be approximated. Second, choosing the projection direction from correlating the input and the output data in Step 2b.1 automatically excludes irrelevant input dimensions. And third, there is no danger of numerical problems due to redundant input dimensions as the univariate regressions can easily be prevented from becoming singular.

Given that we will adjust the distance metric to optimize the local model approximation (see next section), it is also possible to perform LWPR with only *one* projection direction (denoted as LWPR-1). In this case, this distance metric will have to be adjusted to find the optimal receptive field size for a local linearization as well as to make the locally weighted input distribution spherical. An appropriate learning rule of the distance metric can accomplish this adjustment, as explained below. It should be noted that LWPR-1 is obtained from Table 3 by setting $R = 1$.

4.1.2 Adjusting the shape and size of receptive field

The distance metric $\mathbf{D}$ and hence, the locality of the receptive fields, can be learned for each local model individually by stochastic gradient descent in a penalized leave-one-out cross validation cost function (Schaal & Atkeson, 1998):

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i (y_i - \hat{y}_{i,-i})^2 + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{20}$$

where M denotes the number of data points in the training set. The first term of the cost function is the mean leave-one-out cross validation error of the local model (indicated by the subscript $i, -i$) which ensures proper generalization (Schaal & Atkeson, 1998). The second term, the penalty term, makes sure that receptive fields cannot shrink indefinitely in case of large amounts of training data – such shrinkage would be statistically correct for asymptotically unbiased function approximation, but it would require to maintain an ever increasing number of local models in the learning system, which is computationally too expensive. The tradeoff parameter γ can be determined either empirically or from assessments of the maximal local curvature of the function to be approximated (Schaal & Atkeson, 1997); in general, results are not very sensitive to this parameter (Schaal & Atkeson, 1998) as it primarily affects resource efficiency.

It should be noted that due to the *local* cost function in eq.(20), learning becomes entirely localized, too, i.e., no parameters from other local models are needed for updates as, for instance, in competitive learning with mixture models. Moreover, minimizing eq.(20) can be accomplished in an incremental way *without* keeping data in memory (Schaal & Atkeson, 1998). This property is due to a reformulation of the leave-one-out cross validation error as the PRESS residual error (Belsley & Welsch, 1980). As detailed in (Schaal & Atkeson, 1998) the bias-variance tradeoff is thus resolved for every local model *individually* such that an increasing number of local models will not lead to overfitting – indeed, it leads to better approximation results due to model averaging (eq.(19)) in the sense of committee machines(Perrone & Cooper, 1993).

In ordinary weighted linear regression, expanding eq.(20) with the PRESS residual error results in

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (y_i - \hat{y}_i)^2}{(1 - w_i \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2. \tag{21}$$

where $\mathbf{P}$ corresponds to the inverted weighted covariance matrix of the input data. Interestingly, the PRESS residuals of eq.(21) can be *exactly* formulated in terms of the PLS projected inputs

to unit variance – empirically, however, we did not notice a significant improvement of the algorithm, such that we omit this step for the sake of simplicity.

Table 4: Derivatives for distance metric update

For the current data point $\mathbf{x}$*, it's PLS projection* $\mathbf{z}$ *and activation* w*:*
(Refer to Table 3 for some of the variables)

$$\frac{\partial J}{\partial \mathbf{M}} \approx (\sum_{i=1}^{M} \frac{\partial J_1}{\partial w}) \frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W^{n+1}} \frac{\partial J_2}{\partial \mathbf{M}} \text{ (stochastic update of Eq.(22))}$$

$$\frac{\partial w}{\partial M_{kl}} = -\frac{1}{2} w (\mathbf{x}-\mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{kl}} (\mathbf{x}-\mathbf{c}); \quad \frac{\partial J_2}{\partial M_{kl}} = 2\frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij} \frac{\partial D_{ij}}{\partial M_{kl}}$$

$$\frac{\partial D_{ij}}{\partial M_{kl}} = M_{kj}\delta_{il} + M_{ki}\delta_{jl}; \text{ where } \delta_{ij} = 1 \text{ if } i = j \text{ else } \delta_{ij} = 0.$$

$$\sum_{i=1}^{M} \frac{\partial J_1}{\partial w} = \frac{e_{cv}^2}{W^{n+1}} - \frac{2\ e}{W^{n+1}} \mathbf{q}^T \mathbf{a}_H^n - \frac{2}{W^{n+1}} \mathbf{q}^{2T} \mathbf{a}_G^n - \frac{a_E^{n+1}}{(W^{n+1})^2}$$

$$where \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_R \end{bmatrix} \mathbf{z}^2 = \begin{bmatrix} z_1^2 \\ \vdots \\ z_R^2 \end{bmatrix} \mathbf{q} = \begin{bmatrix} z_1/a_{zz,1}^{n+1} \\ \vdots \\ z_R/a_{zz,R}^{n+1} \end{bmatrix} \mathbf{q}^2 = \begin{bmatrix} q_1^2 \\ \vdots \\ q_k^2 \end{bmatrix}$$

$$\mathbf{a}_H^{n+1} = \lambda \mathbf{a}_H^n + \frac{w\ e_{cv}\mathbf{z}}{(1-h)}; \quad \mathbf{a}_G^{n+1} = \lambda \mathbf{a}_G^n + \frac{w^2 e_{cv}^2 \mathbf{z}^2}{(1-h)} \text{ where } h = w\mathbf{z}^T\mathbf{q}$$

$$a_E^{n+1} = \lambda a_E^n + w e_{cv}^2$$

(Refer to Appendix C for the modified $\sum \frac{\partial J_1}{\partial w}$ term for PLS_1)

$\mathbf{z}_i \equiv [z_{i,1} \dots z_{i,R}]^T$ (c.f. Table 3) as

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (y_i - \hat{y}_i)^2}{(1 - w_i \mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \equiv \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} J_1 + \frac{\gamma}{N} J_2 \tag{22}$$

where $\mathbf{P}_z$ corresponds to the covariance matrix computed from the projected inputs $\mathbf{z}_i$ for $R = N$, i.e. the $\mathbf{z}_i$'s spans the same full-rank input space [7] as $\mathbf{x}_i$'s in Eq.(21)(cf. proof in Appendix B). It can also been deduced as explained in Appendix B that $\mathbf{P}_z$ is diagonal, which greatly contributes to the computational efficiency of our update rules. Based on this cost function, the distance metric in LWPR is learned by gradient descent

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \text{ where } \mathbf{D} = \mathbf{M}^T\mathbf{M} \text{ (for positive definiteness)} \tag{23}$$

where $\mathbf{M}$ is an upper triangular matrix resulting from a Cholesky decomposition of $\mathbf{D}$. Following (Schaal & Atkeson, 1998), a stochastic approximation of the gradient $\frac{\partial J}{\partial \mathbf{M}}$ of Eq.22 can be derived by keeping track of several sufficient statistics as shown in Table 4. It should be noted that in these update laws, we treated the PLS projection direction and hence, $\mathbf{z}$ as if it were independent of the distance metric, such that chain rules need not be taken throughout the entire PLS recursions. Empirically, this simplification did not seem to have any negative impact and reduced the update rules significantly. For PLS_1, however, we do provide the complete derivatives in the Appendix C - the only term that changes is the $\sum \frac{\partial J_1}{\partial w}$ term.

[7] For rank deficient input spaces, the equivalence of Eqs.(21) and (22) holds in the space spanned by X

Table 5: Pseudocode of the complete LWPR algorithm

- Initialize the LWPR with no receptive field (RF);
- **For** every new training sample (**x**,y):
  - **For** k=1 to K(# of receptive fields):
    - calculate the activation from eq.(18)
    - update projections & regression (Table 3) and Distance Metric (Table 4)
    - check if number of projections needs to be increased (cf. Section 4.2)
  - **If** no RF was activated by more than w_{gen};
    - create a new RF with $R = 2$, $\mathbf{c} = \mathbf{x}$, $\mathbf{D} = \mathbf{D}_{def}$

4.2 The complete LWPR algorithm

All update rules can be combined in an incremental learning scheme that automatically allocates new locally linear models as needed. The concept of the final learning network is illustrated in Fig. 3 and an outline of the final LWPR algorithm is shown in Table 5.

In this pseudo-code, w_{gen} is a threshold that determines when to create a new receptive field – as discussed in (Schaal & Atkeson, 1998), w_{gen} is a computational efficiency parameter and not a complexity parameter as in mixture models – the closer w_{gen} is set to 1, the more overlap local models will have, which is beneficial in the spirit of committee machines (cf. (Schaal & Atkeson, 1998; Perrone & Cooper, 1993), but more costly to compute – in general, the more overlap is permitted the better the function fitting results, without any danger that the increase in overlap can lead to overfitting. $\mathbf{D}_{def}$ is the initial (usually diagonal) distance metric in eq.(18). The initial number of projections is set to $R = 2$. The algorithm has a simple mechanism of determining whether R should be increased by recursively keeping track of the mean-squared error (MSE) as a function of the number of projections included in a local model, i.e., step 2b.4 in Table 3. If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e., $\frac{MSE_{r+1}}{MSE_r} > \phi$, where $\phi \in [0, 1]$, the algorithm will stop adding new projections locally. As MSE_r can be interpreted as an approximation of the leave-one-out cross validation error of each projection, this threshold criterion avoids problems due to overfitting. Due to the need to compare the MSE of two successive projections, LWPR needs to be initialized with at least two projection dimensions.

4.2.1 Speed-up for learning from trajectories

If in incremental learning, training data is generated from trajectories, i.e., data is temporally correlated, it is possible to accelerate lookup and training times by taking advantage of the the fact that two consecutively arriving training points are close neighbors in input space. For such cases, we added a special data structure to LWPR that allows restricting updates and lookups only to a small fraction of local models instead of exhaustively sweeping through all of them. For this purpose, each local model maintains a list of all other local models that overlap sufficiently with it. Sufficient overlap between two models i and j can be determined from the centers and distance metrics. The point $\mathbf{x}$ in input space that is the closest to both centers in the sense of a Mahalanobis distance is $\mathbf{x} = (\mathbf{D}_i + \mathbf{D}_j)^{-1}(\mathbf{D}_i\mathbf{c}_i + \mathbf{D}_j\mathbf{c}_j)$. Inserting this point into eq.(18) of one of the local models gives the activation w due to this point. The two local models are listed as sufficiently overlapping if $w \geq w_{gen}$ (cf. Table 5). For diagonal distance metrics, the overlap computation is linear in the number of inputs. Whenever a new data point is added to LWPR,

one neighborhood relation is checked for the maximally activated RF. An appropriate counter for each local model ensures that overlap with all other local models is checked exhaustively. Given this "nearest neighbor" data structure, lookup and learning can be confined to only few RFs. For every lookup (update), the identification number of the maximally activated RF is returned. The next lookup (update) will only consider the neighbors of this RF. It can be shown that this method performs as good as an exhaustive lookup (update) strategy that excludes RFs that are activated below a certain threshold w_{cutoff}.

4.2.2 Pruning of local models

As in the RFWR algorithm (Schaal & Atkeson, 1998), it is possible to prune local models depending upon the level of overlap between two local models and/or the accumulcated locally weighted mean-squared error – the pruning strategy is virtually identical as in (Schaal & Atkeson, 1998), Section 3.14. However, due to the numerical robustness of PLS, we have noticed that the need for pruning or merging is almost non-existent in the LWPR implementation, such that we do not expand on this possible feature of the algorithm.

4.2.3 Computational complexity

For a diagonal distance metric $\mathbf{D}$ and under the assumption that the number of projections R remains small and bounded, the computational complexity of one incremental update of all parameters of LWPR is linear in the number of input dimensions N. To the best of our knowledge, this property makes LWPR one of the computationally most efficient algorithms that have been suggested for high dimensional function approximation. This low computational complexity sets LWPR apart from our earlier work on the RFWR algorithm (Schaal & Atkeson, 1998), which was cubic in the number of input dimensions. We thus accomplished one of our main goals, i.e., maintaining the appealing function approximation properties of RFWR while eliminating its problems in high dimensional learning problems.

4.2.4 Confidence Intervals

Under the classical probabilistic interpretation of weighted least squares (Gelman & Rubin, 1995), i.e. that each local model's conditional distribution is normal with heteroscedastic variances $p(y|\mathbf{x}; w_k) \sim N(\mathbf{z}_k{}^T\beta_k, s_k{}^2/w_k)$, it is possible to derive the predictive variances $\sigma^2_{pred,k}$ for a new query point $\mathbf{x}_q$ for each local model in LWPR[8]. The derivation of this measure is in analogy with ordinary linear regression (Schaal & Atkeson, 1994; Myers, 1990) and is also consistent with the Bayesian formulation of predictive variances (Gelman & Rubin, 1995). For each individual local model, $\sigma^2_{pred,k}$ can be estimated as (refer to Table 4 and Table 3 for variable definitions):

$$\sigma^2_{pred,k} \quad = \quad s^2_k(1 + w_k \mathbf{z}^T_{q,k}\mathbf{q}_k) \tag{24}$$

where $\mathbf{z}_{q,k}$ is the projected query point $\mathbf{x}_q$ under the k-th local model, and

$$s_k{}^2 \quad \approx \quad MSE^{n=M}_{k,R}/(M'_k - p'_k); \quad M'_k \equiv \sum_{i=1}^{M} w_{k,i} \approx W_k{}^{n=M}$$

$$p'_k \quad \equiv \quad \sum_{i=1}^{M} w^2_{k,i}\mathbf{z}^T_{k,i}\mathbf{q}_{k,i} \approx a^{n=M}_{p'_k} \quad \text{with incremental update of } \; a^{n+1}_{p'_k} = \lambda a^n_{p'_k} + w_k{}^2 \mathbf{z}_k{}^T \mathbf{q}_k$$

The definition of M' in terms of the sum of weights reflects the effective number of data points entering the computation of the local variance $s_k{}^2$ (Schaal & Atkeson, 1994) after an update of M

[8]Note that w_k is used here as an abbreviated version of $w_{\{q,k\}}$ – the weight contribution due to query point $\mathbf{q}$ in model k– for the sake of simplicity.

training points has been performed. The definition of p', also referred to the as the *local* degrees of freedom, is analogous to the global degrees of freedom of linear smoothers (Hastie & Tibshirani, 1990; Schaal & Atkeson, 1994).

In order to obtain a predictive variance measure for the averaging formula eq.(19), one could just compute the weighed average of the predictive variance in eq.(24). While this approach is viable, it nevertheless ignores important information that can be obtained from variance of the individual predictions $\hat{y}_{q,k}$ and is thus potentially too optimistic. To remedy this issue, we postulate that from the view of combining individual $\hat{y}_{q,k}$, each contributing $y_{q,k}$ was generated from the process

$$y_{q,k} = y_q + \epsilon_1 + \epsilon_{2,k}$$

where we assume two separate noise processes: (i) one whose variance σ^2 is independent of the local model, i.e. $\epsilon_1 \sim N(0, \sigma^2/w_k)$ (and accounts for the differences between the predictions of the local models) and (ii) another, which is the noise process $\epsilon_{2,k} \sim N(0, \sigma^2_{pred,k}/w_k)$ of the individual local models. It can be shown (Appendix D) that eq.(19) is a consistent way of combining prediction from multiple models under the noise model we just described and that the combined predictive variance over all models can be approximated as:

$$\sigma^2_{pred} = \frac{\sum_k w_k \ \sigma^2}{(\sum_k w_k)^2} + \frac{\sum_k w_k \ \sigma^2_{pred,k}}{(\sum_k w_k)^2} \tag{25}$$

The estimate of $\sigma_{pred,k}$ is given in eq.(24). The global variance across models can be approximated as $\sigma^2 = \sum_k w_k (\hat{y}_q - \hat{y}_{k,q})^2 / \sum_k w_k$. Inserting these values in eq.(25), we obtain:

$$\sigma^2_{pred} = \frac{1}{(\sum_k w_k)^2} \sum_{k=1}^{K} w_k [(\hat{y}_q - \hat{y}_{k,q})^2 + s_k^2 (1 + w_k \mathbf{z}_k^T \mathbf{q}_k)] \tag{26}$$

A one-standard-deviation-based confidence interval would thus be

$$I_c = \hat{y}_q \pm \sigma_{pred} \tag{27}$$

The variance estimate in eq.(25) is consistent with the intuitive requirement that when only one local model contributes to the prediction, the variance is entirely attributed to the predictive variance of that single model. Moreover, a query point that does not receive a high weight from any local model will have a large confidence interval due to the small squared sum-of-weight value in the denominator. Figure 4 illustrates comparisons of confidence interval plots on a toy problem with 200 noisy data points. Data from the range [0.5 1.5] was excluded from the training set. Both Gaussian Process Regression and LWPR show qualitatively similar confidence interval bounds and fitting results.

5 Empirical Evaluation

The following sections provide an evaluation of our proposed LWPR learning algorithm over a range of artificial and real world data sets. Whenever useful and feasible, comparisons to state-of-the-art alternative learning algorithms are provided, in particular Support Vector Regression (SVM) and Gaussian Process Regression (GP). SVMR and GPR were chosen due to their generally acknowledged excellent performance in nonlinear regression on finite data sets. However, it should be noted, that both SVM and GP are batch learning systems, while LWPR was implemented as a fully incremental algorithm, as described in the previous sections.

5.1 Function approximation with redundant and irrelevant data

We implemented LWPR algorithm as outlined in Section 4. In each local model, the projection regressions are performed by (locally weighted) PLS, and the distance metric $\mathbf{D}$ is learned by

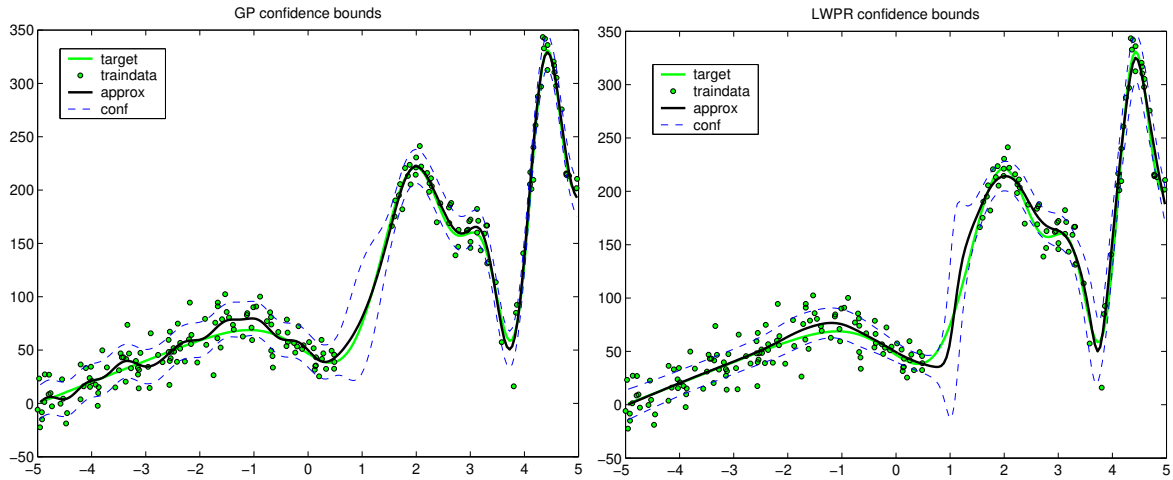


Figure 4: Function approximation with 200 noisy data points along with plots of confidence intervals for (a) Gaussian Process Regression and (b) LWPR algorithms. Note the absence of data in the range [0.5 1.5]

stochastic incremental cross validation; all learning methods employed second order learning techniques, i.e., incremental PLS uses recursive least squares, and gradient descent in the distance metric was accelerated as described in (Schaal & Atkeson, 1998). In all our evaluations, an initial (diagonal) distance metric of $\mathbf{D}_{def} = 30\mathbf{I}$ was chosen, the activation threshold for adding local models was $w_{gen} = 0.2$, and the threshold for adding new projections was $\phi = 0.9$ (cf. Section 4.2). As a first test, we ran LWPR on 500 noisy training data drawn from the two dimensional function (Cross 2D) generated from $y = \max\{\exp(-10x_1^2), \exp(-50x_2^2), 1.25\exp(-5(x_1^2 + x_2^2))\} + \mathrm{N}(0, 0.01)$ as shown in Fig.5(a). This function has a mixture of areas of rather high and rather low curvature and is an interesting test of the learning and generalization capabilities of a learning algorithm: learning models with low complexity find it hard to capture the nonlinearities accurately, while more complex models easily overfit, especially in linear regions. A second test added 8 constant (i.e., redundant) dimensions to the inputs and rotated this new input space by a random 10-dimensional rotation matrix to create a 10 dimensional input space with high rank deficiency (Cross 10D). A third test added another 10 (irrelevant) input dimensions to the inputs of the second test, each having $N(0, 0.05^2)$ Gaussian noise, thus obtaining a data set with 20-dimensional input space (Cross 20D). Typical learning curves with these data sets are illustrated in Fig.5(c). In all three cases, LWPR reduced the normalized mean squared error (thick lines) on a noiseless test set (1681 points on a 41x41 grid in the unit-square in input space) rapidly in 10-20 epochs of training to less than $nMSE = 0.05$, and it converged to the excellent function approximation result of $nMSE = 0.015$ after 100,000 data presentations or 200 epochs[9]. Fig.5(b) illustrates the reconstruction of the original function from the 20-dimensional test data, visualized in 3D – a highly accurate approximation. The rising thin lines in Fig.5(c) show the number of local models that LWPR allocated during learning. The very thin lines at the bottom of the graph indicate the average number of projections that the local models allocated: the average settled at a value of around two local projections, as is appropriate for this originally two dimensional data set. This set of tests demonstrate that LWPR is able to recover a low dimensional nonlinear function embedded in

[9]Since LWPR is an incremental algorithm, data presentations in this case refers to repeated, random order presentations of training data from our noisy data set of size 500

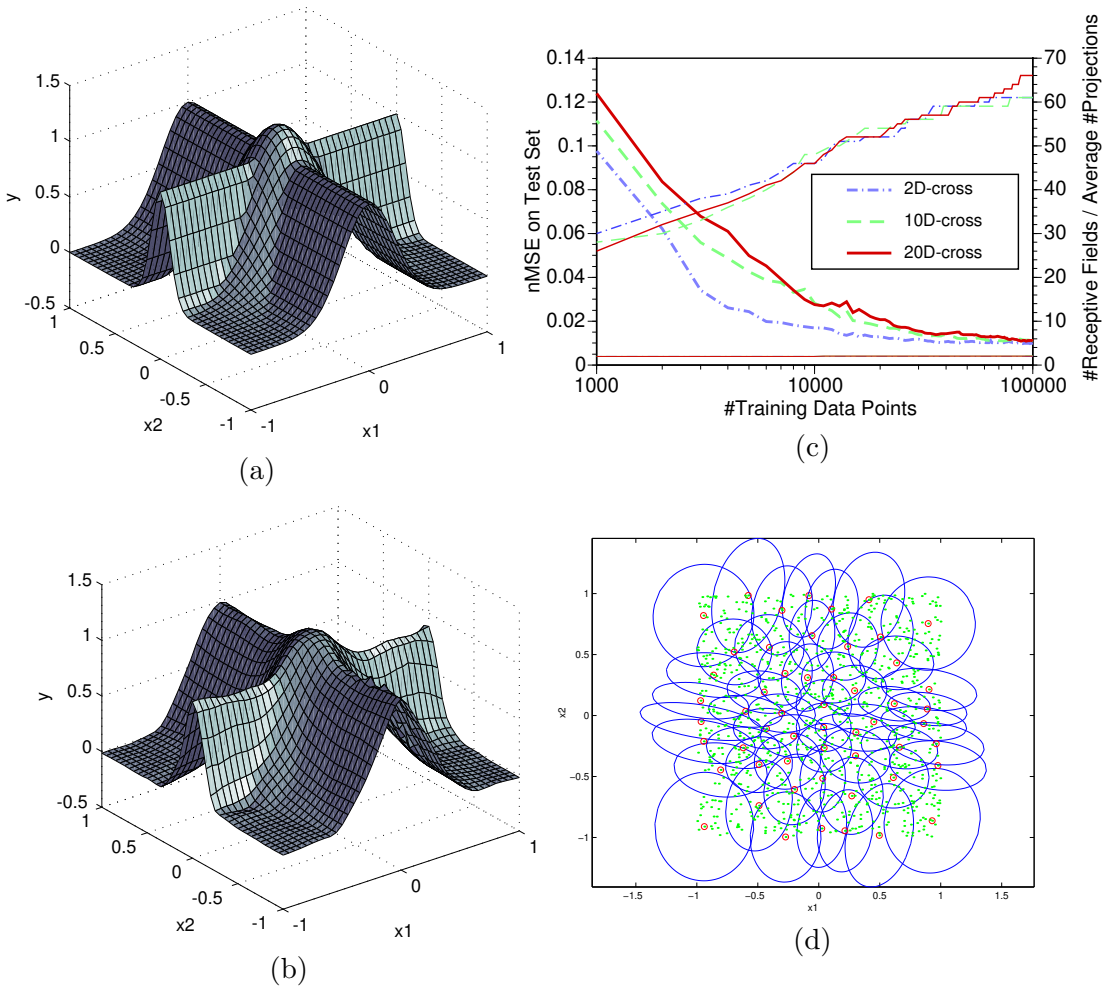


Figure 5: (a) Target and (b) learned nonlinear cross function.(c) Learning curves for 2-D, 10-D and 20-D data (d) The automatically tuned distance metric

high dimensional space despite irrelevant and redundant dimensions, and that the data efficiency of the algorithm does not degrade in higher dimensional input spaces. The computational complexity of the algorithm only increased linearly with the number of input dimensions, as explained in the Section 4.

The results of this evaluations can be directly compared with our earlier work on the RFWR algorithm (Schaal & Atkeson, 1998), in particular Figures 4 and 5 of this earlier paper. The learning speed and the number of allocated local models for LWPR is essentially the same as for RFWR in the 2D test set. Applying RFWR to the 10- and 20-dimensional data set of this paper, however, is problematic, as it requires a careful selection of initial ridge regression parameters to stabilize the highly rank-deficient full covariance matrix of the input data, and it is easy to create too much bias or too little numerical stabilization initially, which can trap the local distance metric adaptation in local minima. While the LWPR algorithm just computes about a factor 10 times longer for the 20D experiment in comparison to the 2D experiment, RFWR requires a 1000-fold increase of computation time, thus rendering this algorithm unsuitable for high-dimensional regression.

In order to compare LWPR's results to other popular regression methods, we evaluated the

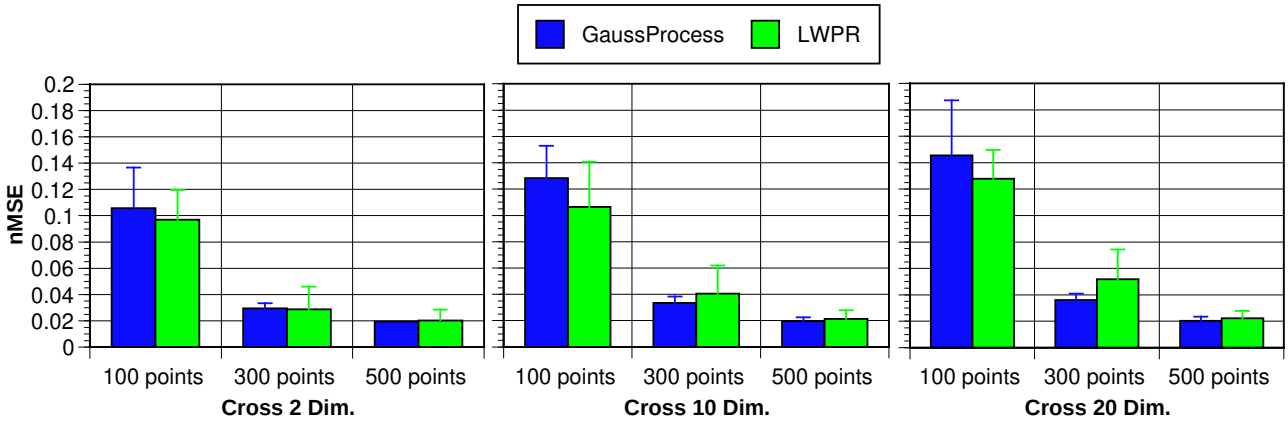


Figure 6: Normalised mean squared error comparisons between LWPR and Gaussian Processes for 2D, 10D and 20D Cross data sets

2D, 10D and 20D cross data sets with Gaussian Process Regression (GP) and Support Vector (SVM) Regression in addition to our LWPR method. It should be noted that neither SVM nor GP methods are incremental methods, although they can be considered the state-of-the-art for batch regression under relatively small number of training data and reasonable input dimensionality. The computational complexity of these methods are prohibitively high for real-time applications. The GP algorithm (Gibbs & Mackay, 1997) used a generic covariance function and optimized over the hyperparameters. The SVM regression was performed using a standard available package (Saunders & Smola, 1998) and optimized for kernel choices.

Fig. 6 compares the performance of LWPR and Gaussian Processes for the above mentioned data sets using 100, 300 and 500 training data points [10]. As in Fig.5 the test data set consisted of 1681 data points corresponding to the vertices of a 41x41 grid over the unit-square; the corresponding output values were the exact function values. The approximation error was measured as a normalized weighted mean squared error, $nMSE$, i.e, the weighted MSE on the test set normalized by the variance of the outputs of the test set – the weights were chosen as $1/\sigma^2_{pred,i}$ for each test point $\mathbf{x}_i$. Using such a weighted $nMSE$ was useful to allow the algorithms to incorporate their confidence in the prediction of a query point, which is especially useful for training data sets with few data points where query points often lie far away from any training data and require strong extrapolation to form a prediction. Multiple runs on 10 randomly chosen training data sets were performed to accumulate the statistics.

As can be seen from Fig. 6, the performance differences of LWPR and GP were largely statistically insignificant across training data sizes and input dimensionality. LWPR had a tendency to perform slightly better on the 100-point data sets, most likely due to its quickly decreasing confidence when significant extrapolation is required for a test point. For the 300-point data sets, GP had a minor advantage and less variance in its predictions, while for 500-point data sets both algorithms achieved equivalent results. While GPs used all the input dimensions for predicting the output (deduced from the final converged coefficients of the covariance matrix), LWPR stopped at an average of 2 local projections reflecting that it exploited the low dimensional distribution of the data. Thus, this comparison illustrates that LWPR is a highly competitive learning algorithm in terms of its generalization capabilities and accuracy of results, inspite of it being a truly incremental, computationally efficient and real-time implementable algorithm.

[10] We have not plotted the results for SVM regression since it was found to consistently perform worse that GP regression for the given number of training data.

Table 6: Comparison of normalized mean squared errors on Boston and Abalone data sets

	Gaussian Process	**Support Vectors**	**LWPR**
Boston	0.0806 ± 0.0195	0.1115 ± 0.09	0.0846 ± 0.0225
Abalone	0.4440 ± 0.0209	0.4830 ± 0.03	0.4056 ± 0.0131

5.2 Comparisons on benchmark regression datasets

While LWPR is specifically geared towards real-time incremental learning in high dimensions, it can nevertheless also be employed for traditional batch data analysis. Here we compare its performance on two natural real world benchmark datasets, using again Gaussian Processes and Support Vector Regression as competitors.

The data sets we used were the Boston Housing data and the Abalone dataset, both available from the UCI Machine Learning Repository (Hettich & Bay, 1999). The Boston housing data, which had 14 attributes, was split randomly (10 random splits) into disjoint sets of 404 training and 102 testing data. The Abalone dataset, which had 9 attributes, was downsampled to yield 10 disjoint sets of 500 training data points and 1177 testing points[11].

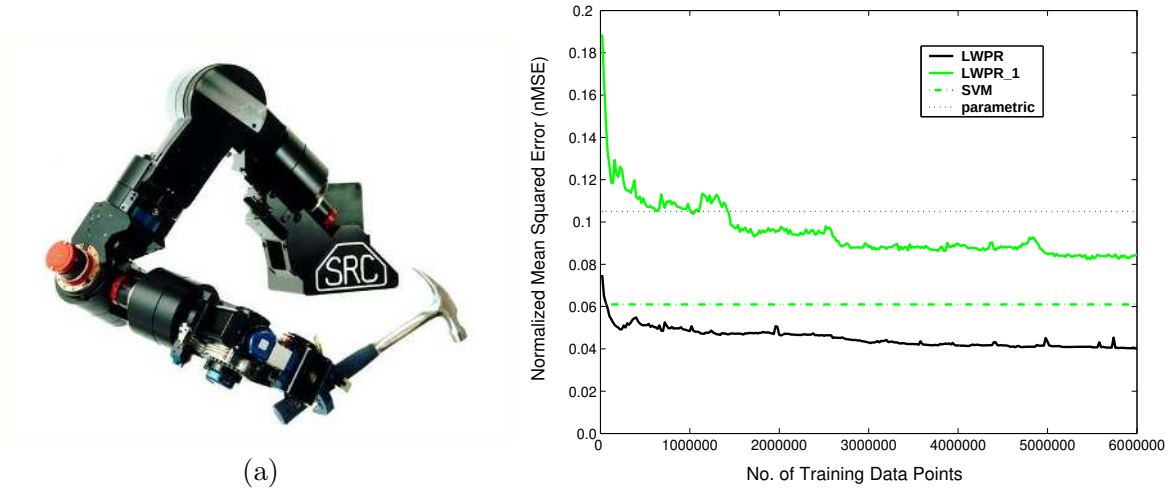
The GP used hyperparameter estimation for the open parameters of the covariance matrix while for SVM regression, the results were obtained by employing a Gaussian kernel of width 3.9 and 10 for the Boston and Abalone datasets, respectively, based on the optimized values suggested in (Scholkopf & Bartlett, 2000). Table 6 shows the comparisons of the normalized mean squared error (nMSE) achieved by GP, SVM and LWPR on both these datasets. Once again, LWPR was highly competitive on these real world data sets, consistently outperforming SVM regression and achieving very similar $nMSE$ results as GP regression.

5.3 Sensorimotor learning in high dimensional space

In this section, we will look at the application of LWPR to real-line learning in high dimensional spaces in a data rich environment – an example of which is learning for robot control. In such domains, LWPR is – to the best of our knowledge – one of the only viable and practical options for principled statistical learning. The goal of learning in this evaluation is to estimate the inverse dynamics model (also referred to as an *internal model*) of the robotic system such that it can be used as a component of a feedforward controller for executing fast, accurate movements.

Before demonstrating the applicability of LWPR in real-time, a comparison with alternative learning methods will serve to demonstrate the complexity of the learning task. We collected 50,000 data points from various movement patterns from a seven degree-of-freedom (DOF) anthropomorphic robot arm (Fig.7a), sampled at 50Hz. The learning task was to fit the the inverse dynamics model of the robot, a mapping from 7 joint positions, 7 joint velocities, and 7 joint accelerations to the corresponding 7 joint torques (i.e, a 21 to 7 dimensional function). 10 percent of this data was excluded from training as a test set. The training data was approximated by four different methods: i) parameter estimation based on an analytical rigid body dynamics model (An & Hollerbach, 1988), ii) Support Vector Regression (Saunders & Smola, 1998) (using a 10-fold downsampled training set for computational feasibility), iii) LWPR-1, i.e., LWPR that only used one single projection (cf. 4.1.1), and iv) full LWPR. It should be noted that neither i) nor ii) are incremental methods. Using a parametric rigid body dynamics model as suggested in i) and just approximating its open parameters from data results in a global model of the inverse dynamics that is theoretically the most powerful method. However, given that our robot is actuated hydraulically

[11]The Gaussian Process algorithm had problems of convergence and numerical stability for training data sizes above 500 points. However, a more comprehensive comparison can be carried out by using techniques from (Williams & Seeger, 2001) to scale up the GP results as pointed out by one of the reviewers



(a) (b)

Figure 7: (a) SARCOS dextrous arm (b) Comparison of nMSE learning curves for learning the robots inverse dynamics model for the shoulder DOF

and rather lightweight and compliant, we know that the rigid body dynamics assumption is not fully justified. In all our evaluations, the inverse dynamics model of each DOF was learned separately, i.e., all models had a univariate output and 21 inputs. LWPR employed a diagonal distance metric.

Fig.7 illustrates the function approximation results for the shoulder motor command graphed over the number of training iterations (one iteration corresponds to the update from one data point). Surprisingly, rigid body parameter estimation achieved the worst results. LWPR-1 outperformed parameter estimation, but fell behind SVM regression. Full LWPR performed the best. The results for all other DOFs were analogous and are not shown here. For the final result, LWPR employed 260 local models, using an average of 3.2 local projections. LWPR-1 did not perform better because we used a diagonal distance metric. The abilities of a diagonal distance metric to "carve out" a locally spherical distribution are too limited to accomplish better results – a full distance metric can remedy this problem, but would make the learning updates quadratic in the number of inputs. As in the previous sections, these results demonstrate that LWPR is a competitive function approximation technique that can be applied successfully in real world applications.

5.3.1 On-line Learning for Humanoid Robots

We implemented LWPR on the real-time operating system (vxWorks) for two of our robotic setups, the seven DOF Sarcos dextrous arm mentioned above in Fig. 7(a), and the Sarcos humanoid robot in Fig. 9(a), a 30 DOF system. Out of the four parallel processors of the system, one 366Mhz PowerPC processor was completely devoted to lookup and learning with LWPR.

For the dexterous arm, each DOF had its own LWPR learning system, resulting in 7 parallel learning modules. In order to accelerate lookup and training times, the nearest neighbor data lookup described in Section 4.2.1 was utilized. The LWPR models were trained on-line while the robot performed a pseudo randomly drifting figure-8 pattern in front of its body. Lookup proceeded at 480Hz, while updating the learning model was achieved at about 70Hz. At 10 second intervals, learning was stopped and the robot attempted to draw a planar figure-8 in the x-z plane of the robot end-effector at 2Hz frequency for the entire pattern. The quality of these drawing patterns is illustrated in Fig. 8. In Fig. 8(a), X_{des} denotes the desired figure-8 pattern, X_{sim} illustrates

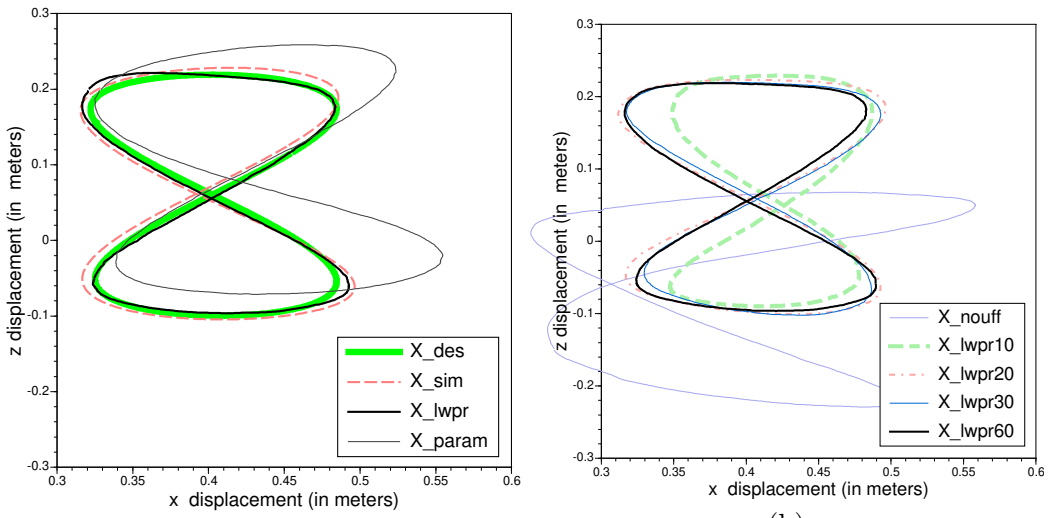
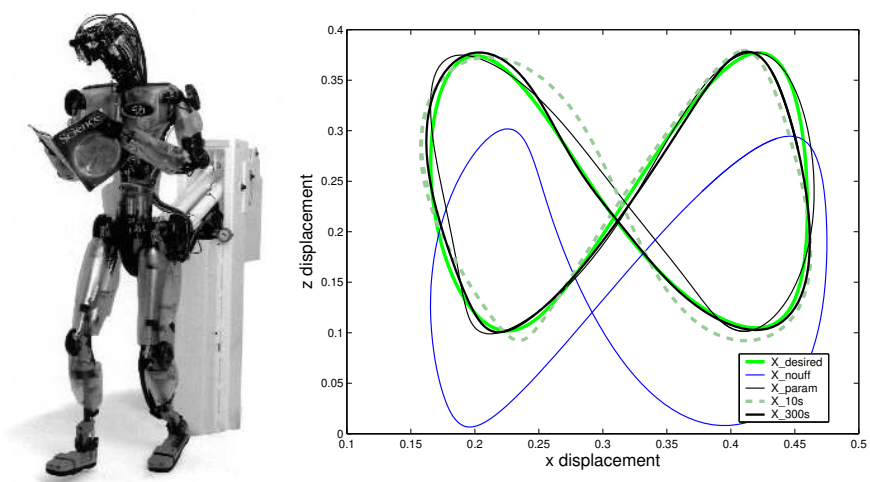


Figure 8: (a) Trajectories of robot end effector: X_{des} is the desired trajectory, X_{sim} a trajectory obtained from a robot simulator that had a perfect controller but numerical inaccuracy due to the integration of the equations of motion, X_{lwpr} is the result of a computed torque controller using LWPR's approximated inverse model, and X_{param} is the trajectory using an inverse model due to rigid body parameter estimation (b) Results of online learning with LWPR starting from scratch, i.e., initially with no functional inverse model – the improvement of control due to LWPR learning is shown at intervals of 10 seconds over the first minute of learning.

the figure-8 performed by our robot simulator that uses a perfect inverse dynamics model (but not necessarily a perfect tracking and numerical integration algorithm), X_{param} is the performance of the estimated rigid body dynamics model, and X_{lwpr} shows the results of LWPR. While the rigid body model has the worst performance, LWPR obtained the best results, even slightly better than the simulator. Fig. 8(b) illustrates the speed of LWPR learning. The X_{nouff} trace demonstrates the figure-8 patterns performed without any inverse dynamics model, just using a low gain PD controller. The other traces show how rapidly LWPR learned the figure-8 pattern during training: they denote performance after 10, 20, 30, and 60 *seconds* of training. After 60 seconds, the figure-8 is hardly distinguishable from the desired trace.

In order to demonstrate the complexity of functions that can be learned in real-time with LWPR, we repeated the same training and evaluation procedure with the Sarcos humanoid robot, which used its right hand to draw a lying figure-8 pattern. In this case, learning of the inverse dynamics model required learning in a 90 dimensional input space, and the outputs where the 30 torque commands for each of the DOFs. As the learning of 30 parallel LWPR models would have exceeded the computational power of our 366Mhz real-time processors, we chose to learn one single LWPR model with a 30 dimensional output vector, i.e., each projection of PLS in LWPR regressed all outputs vs. the projected input data. The projection direction was chosen as the mean projection across all outputs at each projection stage of PLS. This approach is suboptimal, as it is quite unlikely that all output dimensions agree on one good projection direction; essentially, one assumes that the gradients of all outputs point roughly into the same direction. On the other hand, Section2 demonstrated that movement data of actual physical systems lies on locally low dimensional distributions, such that one can hope that LWPR with multiple outputs can still work successfully by simply spanning this locally low dimensional input space with all projections. Fig. 9(b) demonstrates the result of learning in a similar way as Fig. 8(b) – the notation for the different



(a) (b)

Figure 9: (a) The 30-DOF SARCOS humanoid robot (b) Results of online learning of the inverse dynamics with LWPR on the humanoid robot

trajectories in this figure follow as explained above for the 7 DOF robot. Again, LWPR very rapidly improves over a control system with no inverse dynamics controller, i.e., within 10 seconds of movement, the most significant inertial perturbation have been compensated. Convergence to low error tracking of the figure-8 takes slightly longer, i.e., about 300 seconds (X_{300} in Fig. 9(b)), but is reliably achieved. About 50 local models were created for this task. The learned inverse dynamics outperformed a model estimated by rigid body dynamics methods significantly (cf. X_{param} in Fig. 9(b)).

5.3.2 On-line Learning for Autonomous Airplane Control

The on-line learning abilities of LWPR are ideally suited to be incorporated in algorithms of provably stable adaptive control. The control theoretic development of such an approach was presented in Nakanishi et al. (Nakanishi & Schaal, 2004). In essence, the problem formulation begins with a specific class of equations of motion of the form:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\,\mathbf{u} \tag{28}$$

where $\mathbf{x}$ denotes the state of the control system, the control inputs, and $f(\mathbf{x})$ and $g(\mathbf{x})$ are nonlinear function to approximated. A suitable control law for such a system is:

$$\mathbf{u} = \hat{g}(\mathbf{x})^{-1}\left(-\hat{f}(\mathbf{x}) + \dot{\mathbf{x}}_c + \mathbf{K}(\mathbf{x}_c - \mathbf{x})\right) \tag{29}$$

where $\mathbf{x}_c, \dot{\mathbf{x}}_c$ are a desired reference trajectory to be tracked, and the "hat" notation indicates that these are the approximated version of the unknown function.

We applied LWPR in this control framework to learn the unknown function f and g for the problem of autonomous airplane control on a high fidelity simulator. For simplicity, we only considered a planar version of the airplane, governed by the differential equation (Stevens & Lewis, 2003):

$$\begin{aligned} \dot{V} &= \tfrac{1}{m}(T\cos\alpha - D) - g\sin\gamma \\ \dot{\alpha} &= -\tfrac{1}{mV}(L + T\sin\alpha) + \tfrac{g\cos\gamma}{V} + Q \\ \dot{Q} &= cM \end{aligned} \tag{30}$$

In these equations, V denotes the forward speed of the airplane, m the mass, T the thrust, α the angle of attack, g the gravity constant, γ the flight path angle w.r.t. the horizontal world coordinate system axis, Q the pitch rate, and c an inertial constant. The complexity of these equations is hidden in D,L, and M, which are the unknown highly nonlinear aerodynamic drag force, lift force, and pitch moment terms, which are specific to every airplane.

While we will not go into the detail of provably stable adaptive control with LWPR in this paper and how the control law eq.(29) is applied for airplane control, from the viewpoint of learning, the main component to learn are the lift and drag forces, and the pitch moment. These can be obtained by re-arranging eq.(30) to:

$$\begin{aligned} f_D\left(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR}\right) &= T\cos\alpha - \left(\dot{V} + g\sin\gamma\right) m \\ f_L\left(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR}\right) &= \left(\frac{g\cos\gamma}{V} + Q - \dot{\alpha}\right) mV - T\sin\alpha \\ f_M\left(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR}\right) &= \frac{Q}{c}, \end{aligned} \tag{31}$$

where f_D, f_L and f_M refer to the approximated functions of drag (forces), lift (forces) and pitch (moments).
The δ terms denote the control surface angles of the airplane, with indices Midboard-Flap-Left/Right (MFL, MFR), Outboard-Flap-Left/Right (OFL, OFR), and left and right spoilers (SPL, SPR). All terms on the right hand side of eqs.(31) are known, such that we have to cope with three simultaneous function approximation problems in an 11-dimensional input space, an ideal application for LWPR.

We implemented LWPR for the three function above in a high fidelity simulink simulation of an autonomous airplane using the adaptive control approach of (Nakanishi & Schaal, 2004). The airplane started with now initial knowledge, just the proportional controller term in eq.(29) (i.e., the term multiplied by $\mathbf{K}$). The task of the controller was to fly doublets, i.e., up-and-down trajectories, which are essentially sinusoid-like variations of the flight path angle γ

Fig. 10 demonstrates the results of this experiment. Fig. 10(a) shows the desired trajectory in γ and its realization by the controller. Fig. 10(b,c,d) illustrate the on-line function approximation of D, L, and M. As can be seen, the control of γ achieves almost perfect tracking after just very few seconds. The function approximation of D and L very accurate after a very short time. The approximation M requires longer time for convergence, but progresses fast. About 10 local models were needed for learning f_D and f_L, while about 20 local models were allocated for f_M.

An interesting element of Fig. 10 happens after 400 seconds of flight, where we simulated a failure of the airplane mechanics by locking the MFR to 17 degree deflection. As can be seen, the function approximators very quickly re-organize after this change, and the flight is successfully continued, although γ tracking has some error for a while until it converges back to good tracking performance. The strong signal changes in the first seconds after the failure are due to oscillations of the control surfaces, and not a problem in function approximation. Without adaptive control, the airplane would have crashed.

6 Discussion

Nonparametric regression with spatially localized models remains one of the most data-efficient and computationally efficient methods for incremental learning with automatic determination of the model complexity. In order to overcome the curse of dimensionality of local learning systems, this paper investigated methods of linear projection regression and how to employ them in spatially localized nonlinear function approximation for high-dimensional input data that has redundant and irrelevant components. We compared various local dimensionality reduction techniques – an analysis that resulted in choosing a localized version of Partial Least Squares regression at the core of a novel nonparametric function approximator, Locally Weighted Projection Regression

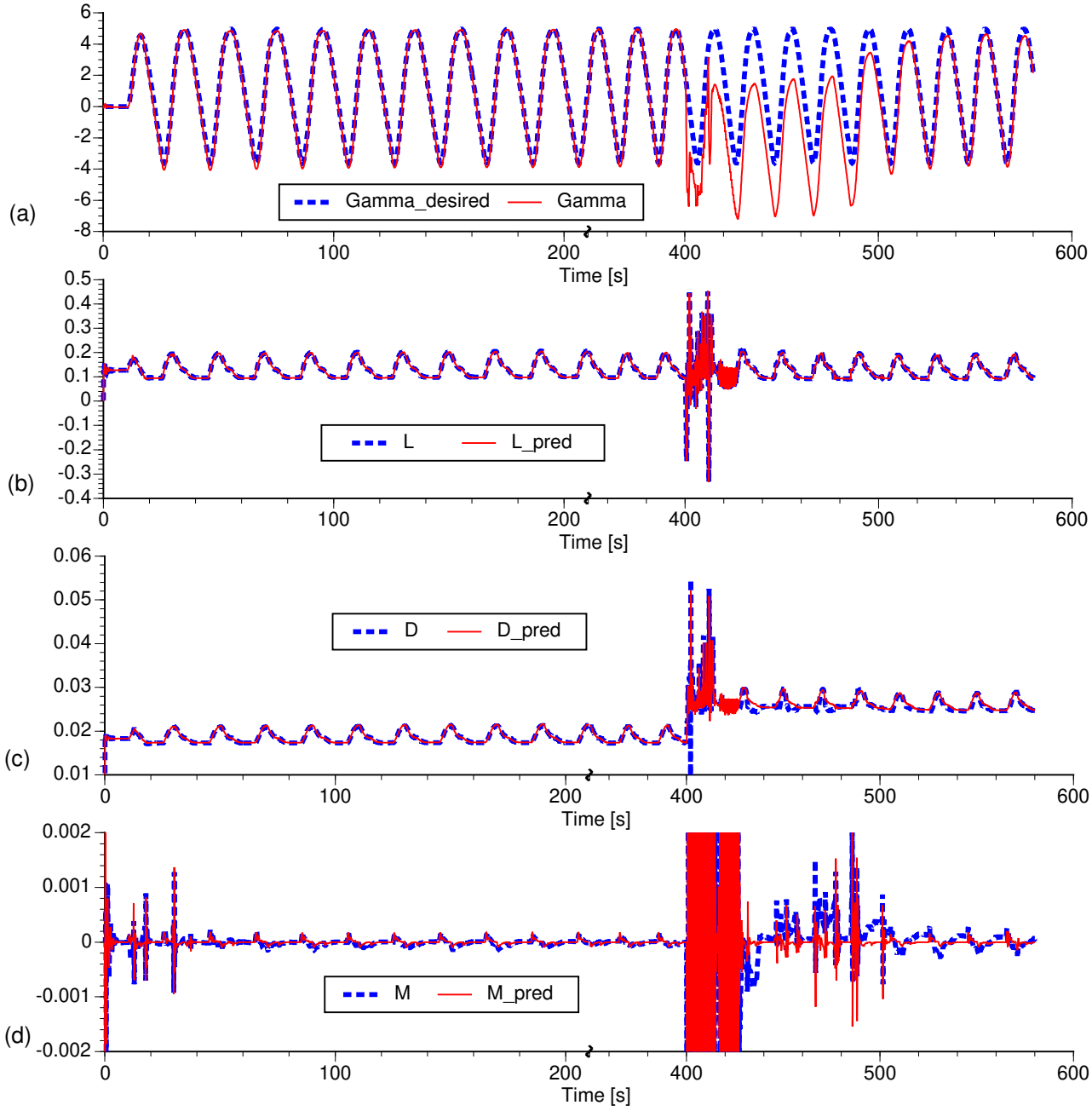


Figure 10: LWPR learning results for adaptive learning control on a simulated autonomous airplane: (a) tracking of flight angle γ, (b) approximation of lift force, (c) approximation of drag force, (d) approximation of pitch moment. At 400 seconds into the flight, a failure is simulated that locks one control surface to a 17 degree angle. Note that for reasons of clearer illustration, an axis break was inserted after 200 seconds.

(LWPR). The proposed technique was evaluated on a range of artificial and real world data sets in up to 50 dimensional input spaces. Besides showing fast and robust learning performance due to second order learning methods based on stochastic leave-one-out cross validation, LWPR exceled by its low computational complexity: updating each local model with a new data point remained *linear* in its computational cost in the number of inputs since the algorithm accomplishes good approximation results with only 3-4 projections irrespective of the number of input dimensions. To our knowledge, this is the first spatially localized incremental learning system that can efficiently work in high dimensional spaces and that is thus suited for on-line and real-time applications. In addition, LWPR compared favorably in its generalization performance with state-of-the-art batch regression regression methods like Gaussian Process Regression, and can provide qualitiatively similar estimates of confidence bounds and predictive variances.

The major drawback of LWPR in its current form is the need for gradient descent to optimize the local distance metrics in each local model, and the manual tuning of a forgetting factor as required in almost all recurisve learning algorithms that accumulate sufficient statistics. Future work will derive a probabilistic version of partial least squares regression that will allow a complete Bayesian treatment of nonparametric regression with locally linear models, hopefully with no remaining open parameters for manual adjustment. Whether a full Bayesian version of LWPR can achieve the same computational efficiency as the nonparametric implemention, however, remains to be seen.

Appendix

A Pseudocode for PLS/PCR/CPR regression

PLS/PCR/CPR Pseudocode (a)

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$
2. **for** $i = 1$ **to** k **do**
   (a) $\mathbf{X}_t = \mathbf{X}_{res}\mathbf{T}$, where $\mathbf{T}$ is a diagonal weight matrix.
   (b) If **[PLS]:** $\mathbf{u}_i = \mathbf{X}_t^T \mathbf{y}_{res}$.
   If **[PCR/CPR]:** $\mathbf{u}_i = [eig\_vec(\mathbf{X}_t^T \mathbf{X}_t)]_{max}$.
   (c) $\beta_i = \mathbf{s}_i^T \mathbf{y}_{res}/(\mathbf{s}_i^T \mathbf{s}_i)$ where $\mathbf{s}_i = \mathbf{X}_{res}\mathbf{u}_i$.
   (d) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{s}_i\beta_i$.
   (e) $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{s}_i\mathbf{p}_i{}^T$ where $\mathbf{p}_i = \mathbf{X}_{res}^T\mathbf{s}_i/(\mathbf{s}_i^T\mathbf{s}_i)$.

   where $\mathbf{T} = \mathbf{I}$ for PLS and PCR.

Locally Weighted version of PLS/PCR/CPR regression (b)

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$
2. **for** $i = 1$ **to** k **do**
   (a) $\mathbf{X}_t = \mathbf{X}_{res}\mathbf{T}$, where $\mathbf{T}$ is a diagonal weight matrix.
   (b) If **[PLS]:** $\mathbf{u}_i = \mathbf{W}\mathbf{X}_t^T \mathbf{y}_{res}$.
   If **[PCR/CPR]:** $\mathbf{u}_i = [eig\_vec(\mathbf{X}_t^T \mathbf{W}\mathbf{X}_t)]_{max}$.
   (c) $\beta_i = \mathbf{W}\mathbf{s}_i^T \mathbf{y}_{res}/(\mathbf{s}_i^T \mathbf{W}\mathbf{s}_i)$ where $\mathbf{s}_i = \mathbf{X}_{res}\mathbf{u}_i$.
   (d) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{s}_i\beta_i$.
   (e) $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{s}_i\mathbf{p}_i{}^T$ where $\mathbf{p}_i = \mathbf{W}\mathbf{X}_{res}^T\mathbf{s}_i/(\mathbf{s}_i^T\mathbf{W}\mathbf{s}_i)$.

B PRESS residuals for PLS

We prove that, under the assumption that $\mathbf{x}$ lives in a reduced dimensional subspace, the PRESS residuals of Eq.(21) can indeed be replaced by the residual denoted by Eq.(22), i.e.

$$\mathbf{x}_i^T \mathbf{P} \mathbf{x}_i = \mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i \tag{32}$$

where $\mathbf{P} = (\mathbf{X}^T\mathbf{X})^{-1}$, $\mathbf{P}_z = (\mathbf{Z}^T\mathbf{Z})^{-1}$ corresponds to the inverse covariance matrices.
(Refer to Table 1 for the batch notations.)
Part 1: Let $\mathbf{T}$ be the transformation matrix with full rank in row space that denotes coordinate transformation from the rank deficient space of $\mathbf{x}$ to the full rank space of $\mathbf{z}$. Then, for any $\mathbf{z} = \mathbf{T}^T\mathbf{x}$ and the corresponding inverse covariance matrix $\mathbf{P}_z$, we can show that

$$\mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i = \mathbf{x}_i^T \mathbf{T}((\mathbf{X}\mathbf{T})^T(\mathbf{X}\mathbf{T}))^{-1}\mathbf{T}^T\mathbf{x}_i = \mathbf{x}_i^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i = \mathbf{x}_i^T\mathbf{P}\mathbf{x}_i \tag{33}$$

i.e., a linear transformation maintains the norm.
Part 2: In this part, we will show that the recursive PLS projections which transform the inputs $\mathbf{x}$ to $\mathbf{z}$ can be written as a linear transformation matrix, which completes our proof. Clarifying some of the notation:

$$\mathbf{X} \equiv [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_M]^T, \quad \mathbf{Z} \equiv [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_M]^T \equiv [\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_k]. \tag{34}$$

We will now look at each of the k PLS projection directions and attempt to show that $\mathbf{z}_i = \mathbf{T}^T\mathbf{x}_i$ or (in a batch sense) $\mathbf{Z} = \mathbf{X}\mathbf{T}$ by showing (for each individual projections) k, there exists a $\mathbf{t}_k$ such that:

$$\mathbf{s}_k = \mathbf{X}\mathbf{t}_k, \quad \textit{where} \quad \mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_k]. \tag{35}$$

For $k = 1$ (c.f. Table 3),

$$\mathbf{s}_1 = \mathbf{X}\mathbf{X}^T\mathbf{y} = \mathbf{X}\mathbf{t}_1, \quad \textit{where} \quad \mathbf{t}_1 = \mathbf{X}^T\mathbf{y}. \tag{36}$$

For $k = 2$ (c.f. Table 3),

$$\mathbf{s}_2 = \mathbf{X}_{res}\mathbf{X}_{res}^T\mathbf{y}_{res} \tag{37}$$

We also know from the algorithm (c.f. Table 3) that,

$$\mathbf{X}_{res} = \mathbf{X} - \frac{\mathbf{s}_1\mathbf{s}_1^T\mathbf{X}}{\mathbf{s}_1^T\mathbf{s}_1} = (\mathbf{I} - \frac{\mathbf{s}_1\mathbf{s}_1^T}{\mathbf{s}_1^T\mathbf{s}_1})\mathbf{X} = P_{\mathbf{s}_1}\mathbf{X} \tag{38}$$

$$\mathbf{y}_{res} = \mathbf{y} - \frac{\mathbf{s}_1\mathbf{s}_1^T\mathbf{y}}{\mathbf{s}_1^T\mathbf{s}_1} = (\mathbf{I} - \frac{\mathbf{s}_1\mathbf{s}_1^T}{\mathbf{s}_1^T\mathbf{s}_1})\mathbf{y} = P_{\mathbf{s}_1}\mathbf{y} \tag{39}$$

where $P_{\mathbf{s}_1}$ represents a projection operator. Using results from Eq.(38) and Eq.(39) in Eq.(37),

$$\begin{aligned} \mathbf{s}_2 &= P_{\mathbf{s}_1}\mathbf{X}(P_{\mathbf{s}_1}\mathbf{X})^T P_{\mathbf{s}_1}\mathbf{y} \\ &= P_{\mathbf{s}_1}\mathbf{X}\mathbf{X}^T P_{\mathbf{s}_1}\mathbf{y} \ \ldots \text{using property of projection operator: } P_{\mathbf{s}_1} = P_{\mathbf{s}_1}^T = P_{\mathbf{s}_1}P_{\mathbf{s}_1} \\ &= P_{\mathbf{s}_1}\mathbf{X}\mathbf{t}_2' \end{aligned} \tag{40}$$

It can be shown easily by writing out the psuedo inversion that there exists an operator $\mathbf{R}_2 = (\mathbf{I} - \mathbf{u}_1\mathbf{u}_1^t\mathbf{X}^T\mathbf{X}/\mathbf{u}_1^T\mathbf{X}^T\mathbf{X}\mathbf{u}_1)$ such that

$$P_{\mathbf{s}_1}\mathbf{X} = \mathbf{X}\mathbf{R}_2 \tag{41}$$

Using Eqs.(40) and (41), we can write

$$\mathbf{s}_2 = \mathbf{X}\mathbf{t}_2 \ \ \textit{where}\ \mathbf{t}_2 = \mathbf{R}_2\mathbf{t}_2' = (I - \frac{\mathbf{u}_1\mathbf{u}_1^t\mathbf{X}^T\mathbf{X}}{\mathbf{u}_1^T\mathbf{X}^T\mathbf{X}\mathbf{u}_1})\mathbf{X}^T P_{\mathbf{s}_1}\mathbf{y}. \tag{42}$$

This operation can be carried out recursively to determine all the $\mathbf{t}_k$, showing that the PLS projections can be written as a linear transformation. This completes the proof of the validity of the modified PRESS residual of Eq.(32) for PLS projections.

Also note that, $\mathbf{P}_z$ is diagonal by virtue of the fact that in the PLS algorithm, after every projection iteration, the projected components of input space $\mathbf{X}$ is subtracted before computing the next projection (Table 1(c) or Table 3(2b.3)), ensuring the next component of $\mathbf{Z}$ will always be orthogonal to the previous ones. This property was discussed in (Frank, 1993).

C Distance Metric Update for PLS_1

$$\begin{aligned}
\sum_{i=1}^{M} \frac{\partial J_1}{\partial w} &= \frac{e_{cv}^2}{W^{n+1}} - 2\frac{z^{(1)}\ e}{W^{n+1} a_{zz,1}^{n+1}} a_H^n - 2\frac{{z^{(1)}}^2}{W^{n+1}(a_{zz,1}^{n+1})^2} a_R^n - \frac{a_E^{n+1}}{(W^{n+1})^2} \\
&\quad + [\mathbf{a}_T^{n+1} - 2\frac{a_R^{n+1}\mathbf{a}_C^{n+1}}{a_{zz,1}^{n+1}}] \frac{(\mathbf{I} - \mathbf{u}_1\mathbf{u}_1^T/(\mathbf{u}_1^T\mathbf{u}_1))\mathbf{x}\ res_1}{W^{n+1}\sqrt{\mathbf{u}_1^T\mathbf{u}_1}} \\
\mathbf{a}_C^{n+1} &= \lambda \mathbf{a}_C^n + w z^{(1)} \mathbf{x}^T; \quad a_H^{n+1} = \lambda H^n + \frac{w\ e_{cv} z^{(1)}}{(1-h)} \\
a_R^{n+1} &= \lambda a_R^n + \frac{w^2 {z^{(1)}}^2 e_{cv}^2}{(1-h)} \text{ where } h = \frac{w\ {z^{(1)}}^2}{a_{zz,1}^{n+1}} \\
a_E^{n+1} &= \lambda a_E^n + w e_{cv}^2; \quad \mathbf{a}_T^{n+1} = \lambda \mathbf{a}_T^n + \frac{w(2\ w\ e_{cv}^2 z^{(1)}/a_{zz,1}^{n+1} - e_{cv}\beta_1^{n+1})}{(1-h)} \mathbf{x}^T
\end{aligned}$$

D Combined predictive variances

The noise model for combining prediction from individual local model is:

$$y_{q,k} = y_q + \epsilon_1 + \epsilon_{2,k}$$

where $\epsilon_1 \sim N(0, \sigma^2/w_k)$ and $\epsilon_{2,k} \sim N(0, \sigma_{pred,k}^2/w_k)$. The mean prediction due to multiple local models can be written according to a heterscedastic average:

$$\hat{y}_q = \sum_k \left(\frac{w_k}{\sigma^2 + \sigma_{pred,k}^2} y_{q,k}\right) / \sum_k \left(\frac{w_k}{\sigma^2 + \sigma_{pred,k}^2}\right) \approx \frac{\sum_k w_k y_{q,k}}{\sum_k w_k} \approx \frac{\sum_k w_k \hat{y}_{q,k}}{\sum_k w_k} \tag{43}$$

under the assumption that $(\sigma^2 + \sigma_{pred,k}^2)$ is approximately constant for all contributing models k and that $\hat{y}_{q,k}$ is an estimate over multiple noisy instances of $y_{q,k}$ that has averaged out the noise process $\epsilon_{2,k}$ – exactly what happens within each local model. Thus, eq.(43) is consistent with eq.(19) under the proposed dual noise model. The combined predictive variance can now be derived as:

$$\begin{aligned}
\sigma_{pred}^2 &= E\{y_q^2\} - (E\{y_q\})^2 = E\{(\frac{\sum_k w_k y_{q,k}}{\sum_k w_k})^2\} - (E\{y_q\})^2 \\
&= \frac{1}{(\sum_k w_k)^2} E\{(\sum_k w_k y_q)^2 + (\sum_k w_k \epsilon_1)^2 + (\sum_k w_k \epsilon_{2,k})^2\} - (\hat{y}_q)^2 \\
&= \frac{1}{(\sum_k w_k)^2} E\{(\sum_k w_k \epsilon_1)^2 + (\sum_k w_k \epsilon_{2,k})^2\}
\end{aligned} \tag{44}$$

Using the fact that $E\{\mathbf{x}^2\} = (E\{\mathbf{x}\})^2 + var(\mathbf{x})$ and noting that ϵ_1 and $\epsilon_{2,k}$ have zero mean,

$$\begin{aligned}
\sigma^2_{pred} &= \frac{1}{(\sum_k w_k)^2} var(\sum_k w_k \epsilon_1) + \frac{1}{(\sum_k w_k)^2} var(\sum_k w_k \epsilon_{2,k}) \\
&= \frac{1}{(\sum_k w_k)^2} [\sum_k w_k^2 \frac{\sigma^2}{w_k}] + \frac{1}{(\sum_k w_k)^2} [\sum_k w_k^2 \frac{\sigma^2_{pred,k}}{w_k}] \\
\sigma^2_{pred} &= \frac{\sum_k w_k \sigma^2}{(\sum_k w_k)^2} + \frac{\sum_k w_k \sigma^2_{pred,k}}{(\sum_k w_k)^2} \qquad (45)
\end{aligned}$$

which gives the expression for the combined predictive variances.

E References

An, C. H., A. C., & Hollerbach, J. (1988). *Model based control of a robot manipulator*. MIT Press.

Atkeson, C., Moore, A., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*(4), 76–113.

Bell, A., & Sejnowski, T. (1997). The Independent Components of natural scenes are edge filters. *Vision Research*, *37*(23), 3327–3338.

Belsley, D. A., K. E., & Welsch, D. (1980). *Regression diagnostics*. John Wiley.

D'Souza, A., V. S., & Schaal, S. (2001). Are internal models of the entire body learnable? *Society for Neuroscience Abstracts*, Vol. 27.

Everitt, B. S. (1984). *An introduction to latent variable models*. Chapman and Hall, London.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (pp. 524–532). Morgan-Kaufmann, Los Altos CA.

Frank, I. E., a. (1993). A statistical view of some chemometric regression tools. *Technometrics*, *35*(2), 109–135.

Friedman, J. H., & Stutzle, W. (1981). Projection pursuit regression. *Journal of America. Statistical Association*, *76*, 817–823.

Gelman, A. B., C. J. S. S. H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall.

Ghahramani, Z., & Beal, M. (2000). Variational inference for bayesian mixtures of factor analysers. In K. M. T.K. Leen,S. A. Solla (Ed.), *Advances in Neural Information Processing Systems 12* (pp. 449–455). MIT Press.

Gibbs, M., & Mackay, D. J. C. (1997). *Efficient implementation of gaussian processes* (Technical report). Cavendish Laboratory, Cambridge, UK.

Hastie, T., & Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, *8*, 120–143.

Hastie, T., & Tibshirani, R. (1990). *Generalized additive models*. Chapman and Hall, London.

Hettich, S., & Bay, S. (1999). The uci kdd archive. Univeristy of California, Irvine, Department of Information and Computer Science.

Horn, R., & Johnson, C. (1994). *Matrix analysis*. Press Syndicate of the University of Cambridge.

Jordan, M., & Jacobs, R. (1994). Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, *6*(2), 181–214.

Kawato, M. (1999). Internal models fot motor control and trajectory planning. *Current Opinions in Neurobiology*, *9*, 718–727.

Ljung, L., & Soderstrom, T. (1986). *Theory and prcatice of recursive identification*. MIT Press.

Massy, W. F. (1965). Pricipal component regression in exploratory statistical research. *Jornal of the American statistical Association*, *60*, 234–246.

Myers, R. H. (1990). *Classical and modern regression with applications*. PWS - Kent.

Nakanishi, J., F. J. A., & Schaal, S. (2004). Composite adaptive control with locally weighted statistical learning. *IEEE International Conference on Robotics and Automation* (pp. 2647–2652).

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive eld properties by learning a sparse code for natural images. *Nature*, *381*, 607–609.

Perrone, M. P., & Cooper, L. N. (1993). *Neural networks for speech and image processing*, Chap. When Networks disagree: Ensemble methods for hybrid neural networks. Chapman Hall.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by local linear embedding. *Science*, *290*, 2323–2326.

Rubin, D., & Thayer, D. (1982). EM algorithms for ML factor analysis. *Psychometrika*, *47*(1), 69–76.

Sanger, T. (1989). Optimal unsupervised learning in a single layer feedforward neural network. *Neural Networks*, *2*, 459–473.

Saunders, C., S. M. W. J. B. L. B., & Smola, A. (1998). *Support vector machine - reference manual* (Technical Report TR CSD-TR-98-03). Dept. of Computer Science, Royal Holloway, Univ. of London.

Schaal, S., & Atkeson, C. (1994). Assessing the quality of learned local models. *Advances in Neural Information Processing Systems 6* (pp. 160–167). Morgan-Kaufmann.

Schaal, S., & Atkeson, C. (1997). *Receptive field weighted regression* (Technical report tr-h-209). ATR Human Information Processing, Kyoto, Japan.

Schaal, S., & Atkeson, C. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*(8), 2047–2084.

Schaal, S., & Sternad, D. (2001). Origins and violoations of the 2/3 power law in rhythmic 3d movements. *Experimental Brain Research*, *136*, 60–72.

Scholkopf, B., B. C., & Smola, A. J. (1999). *Advances in kernel methods: Support vector learning*. MIT Press.

Scholkopf, B., S. A. J. W. R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, *12*(5), 1207–1245.

Scott, D. (1992). *Multivariate density estimation*. Wiley-NY.

Smola, A. J., & Scholkopf, B. (1998). *A tutorial on support vector regression* (NEUROCOLT Technical Report NC-TR-98-030). Royal Holloway College, London.

Stevens, B. L., & Lewis, F. L. (2003). *Aircraft control and simulation*. John Wiley, New Jersey.

Tenenbaum, J., d. S. V., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Tipping, M., & Bishop, C. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B*, *61*(3), 611–622.

Ueda, N., N. R. G. Z., & Hinton, G. (2000). SMEM algorithm for mixture models. *Neural Computation*, *12*, 2109–2128.

Vijayakumar, S., & Ogawa, H. (1999). RKHS based functional analysis for exact incremental learning. *Neurocomputing*, *29*(1–3), 85–113.

Vijayakumar, S., & Schaal, S. (1998). Local Adaptive Subspace Regression. *Neural Processing Letters*, *7*(3), 139–149.

Vijayakumar, S., & Schaal, S. (2000). O(n) algorithm for incremental real time learning in high dimensional space. *International Conference on Machine Learning (ICML)* (pp. 1079–1086).

Vlassis, N., M. Y. K. B. (2002). Supervised dimensionality reduction of intrinsically low-dimensional data. *Neural Computation*.

Williams, C. K. I., & Rasmussen, C. (1996). Gaussian processes for regression. In M. Touretzky, & Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*. MIT Press.

Williams, C. K. I., & Seeger, M. (2001). Using the Nystrom method to speedup kernel machines. *Advances in Neural Information Processing Systems 13*. MIT Press.

Wold, H. (1975). *Perspectives in probability and statistics*, Chap. Soft modelling by latent variables: the nonlinear iterative partial least squares approach. Chapman Hall.

Xu, L., Jordan, M., & Hinton, G. (1995). An alternative model for mixtures of experts. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in Neural Information Processing Systems*, Vol. 7 (pp. 633–640). The MIT Press.